

EXPRESS MAIL MAILING LABEL NUMBER EL315197625US

PATENT

10830.0081.NPUS00

EMC-01-076

APPLICATION FOR UNITED STATES LETTERS PATENT

for

INSERTION OF NOISE FOR REDUCTION IN THE NUMBER OF BITS FOR  
VARIABLE-LENGTH CODING OF (RUN, LEVEL) PAIRS

by

Sorin Faibish

Ugur Sezer

Seyfullah H. Oguz

Wayne W. Duso

DRAFTING UNIT - PCT

1 BACKGROUND OF THE INVENTION

2 1. Field of the Invention

3 The present invention relates to variable-length coding for data compression, and  
4 in particular the variable-length coding of (run, level) pairs.

5 2. Background Art

6 Variable-length coding (VLC) is a well-known technique of data compression. If  
7 certain data patterns occur much more often than other data patterns, then data  
8 compression occurs if the frequent data patterns are assigned shorter variable-length  
9 codes, and the infrequent data patterns are assigned longer variable-length codes. The  
10 technique has been well known since the introduction of the Morse code for telegraphy in  
11 the 19<sup>th</sup> century. The Morse code, for example, is a set of variable-length code words,  
12 each of which is comprised of a series of “dots” and “dashes.” The most frequently used  
13 letters of the alphabet (E and T) are assigned the shortest code words, consisting of a  
14 single dot for the letter E, and a single dash for the letter T.

15 For encoding pictures or audio-visual data, the variable-length coding often  
16 encodes (run, level) pairs. Typically the run represents the number of consecutive zero-  
17 valued transform coefficients in a series of transform coefficients, and the level  
18 represents the nonzero value of a transform coefficient which terminates the above chain  
19 of zero valued coefficients. For example, a number of standard image compression  
20 techniques subdivide a picture into 8x8 blocks of pixels. For each block, a two-  
21 dimensional discrete cosine transform is applied to the pixel values to produce a series of  
22 64 DCT coefficients. The coefficient values are quantized in such a way that a large

1 number of the coefficients typically have a level of zero, and most of the coefficients  
2 have relatively small level magnitudes.

3 There are various ways that (run, level) pairs can be variable-length coded. In a  
4 common coding scheme, each of the most frequent (run, level) pairs is assigned a  
5 corresponding variable-length code having a length that is inversely proportional to the  
6 frequency of the (run, level) pair. The less frequent (run, level) pairs, however, are  
7 encoded in a different fashion, which will be referred to as an escape sequence. For  
8 example, the escape sequence is a fixed-length code word, consisting of an escape code,  
9 followed by the run value and the level value. Such variable-length coding of (run, level)  
10 pairs is used in a popular video compression technique known as MPEG.

11 MPEG is an acronym for the Moving Picture Experts Group, which was set up by  
12 the International Standards Organization (ISO) to work on compression of video and its  
13 associated audio. MPEG provides a number of different variations (MPEG-1, MPEG-2,  
14 etc.) to suit different bandwidth and quality constraints. MPEG-2, for example, is  
15 especially suited to the storage and transmission of broadcast quality television programs.

16 For the video data, MPEG provides a high degree of compression (up to 200:1) by  
17 encoding 8 x 8 blocks of pixels into a set of discrete cosine transform (DCT) coefficients,  
18 quantizing and encoding the coefficients, and using motion compensation techniques to  
19 encode most video frames as predictions from or between other frames. In particular, the  
20 encoded MPEG video stream is comprised of a series of groups of pictures (GOPs), and  
21 each GOP begins with an independently encoded (intra) I frame and may include one or  
22 more following P frames and B frames. Each I frame can be decoded without  
23 information from any preceding and/or following frame. Decoding of a P frame in

1 general requires information from a preceding frame in the same GOP. Decoding of a B  
2 frame in general requires information from both a preceding frame which can be in the  
3 previous or the same GOP, and a following frame in the same GOP. To minimize  
4 decoder buffer requirements, transmission orders differ from presentation orders for some  
5 frames, so that all the information of the reference frames required for decoding a non-  
6 causally predicted frame, that is to say a B frame will arrive at the decoder before the B  
7 frame.

8 In addition to the motion compensation techniques for video compression, the  
9 MPEG standard provides a generic framework for combining one or more elementary  
10 streams of digital video and audio, as well as system data, into single or multiple program  
11 transport streams (TS) which are suitable for storage or transmission. The system data  
12 includes information about synchronization, random access, management of buffers to  
13 prevent overflow and underflow, and time stamps for video frames and audio packetized  
14 elementary stream packets embedded in video and audio elementary streams as well as  
15 program description, conditional access and network related information carried in other  
16 independent elementary streams. The standard specifies the organization of the  
17 elementary streams and the transport streams, and imposes constraints to enable  
18 synchronized decoding from the audio and video decoding buffers under various  
19 conditions.

20 The MPEG-2 standard is documented in ISO/IEC International Standard (IS)  
21 13818-1, "Information Technology-Generic Coding of Moving Pictures and Associated  
22 Audio Information: Systems," ISO/IEC IS 13818-2, "Information Technology-Generic  
23 Coding of Moving Pictures and Associated Audio Information: Video," and ISO/IEC IS

1 13818-3, "Information Technology-Generic Coding of Moving Pictures and Associated  
2 Audio Information: Audio," which are incorporated herein by reference. A concise  
3 introduction to MPEG is given in "A guide to MPEG Fundamentals and Protocol  
4 Analysis (Including DVB and ATSC)," Tektronix Inc., 1997, incorporated herein by  
5 reference.

6

7 **SUMMARY OF THE INVENTION**

8 The present invention recognizes an opportunity for obtaining a significant  
9 reduction in the total bit count for variable-length coding of (run, level) pairs by the  
10 introduction of an insignificant amount of noise into the information being encoded.  
11 There are circumstances that indicate such an opportunity. The structure of the variable-  
12 length coding scheme may inherently provide such an opportunity for certain (run, level)  
13 pairs. The presence of an escape mechanism in the coding scheme may provide an  
14 opportunity for obtaining a significant reduction in the total bit count for variable-length  
15 coding if the introduction of a small amount of noise eliminates an escape sequence. In  
16 addition, the scheme for variable-length coding of the (run, level) pairs may have been  
17 especially designed for certain nominal statistics of the (run, level) pairs. In this case, if  
18 the statistics of the (run, level) pairs for a particular instance deviate from the expected  
19 statistics, there may be an opportunity for a significant reduction in total bit count. There  
20 may also be an opportunity for a significant reduction in total bit count if the statistics of  
21 the (run, level) pairs in general have been modified due to a change from the way that the  
22 (run, level) pairs are normally produced from the source material. For example, there

1 may be an opportunity if there has been a change causing the average run length to  
2 increase.

3 In accordance with one aspect of the present invention, there is provided a method  
4 of processing information represented by an original series of (run, level) pairs. The  
5 method includes inspecting the (run, level) pairs in the original series of (run, level) pairs  
6 to determine whether or not modification of at least one (run, level) pair in the original  
7 series of (run, level) pairs would produce a desirable decrease in a number of bits  
8 required for variable-length encoding of the information despite introduction of noise into  
9 the variable-length encoding of the information. Upon determining that modification of  
10 at least one (run, level) pair in the original series of (run, level) pairs would produce a  
11 desirable decrease in the number of bits required for variable-length encoding of the  
12 information despite introduction of noise into the variable-length encoding of the  
13 information, the at least one (run, level) pair is modified to produce a modified series of  
14 (run, level) pairs from the original series of (run, level) pairs. The method further  
15 includes variable-length encoding the modified series of (run, level) pairs.

16 In accordance with another aspect, the invention provides a method of variable-  
17 length encoding a block of pixels. The method includes computing a two-dimensional  
18 discrete cosine transform (DCT) of the block of pixels to produce a series of DCT  
19 coefficient values, quantizing the DCT coefficient values to produce quantized  
20 coefficient values, and producing an original series of (run, level) pairs each having a  
21 level value indicating a respective non-zero quantized coefficient value. The method  
22 further includes inspecting the (run, level) pairs in the original series of (run, level) pairs  
23 to determine whether or not modification of at least one (run, level) pair in the original

1 series of (run, level) pairs would produce a desirable decrease in a number of bits  
2 required for variable-length encoding of the block of pixels despite introduction of noise  
3 into the variable-length encoding of the block of pixels. Upon determining that  
4 modification of at least one (run, level) pair in the original series of (run, level) pairs  
5 would produce a desirable decrease in the number of bits required for variable-length  
6 encoding of the block of pixels despite introduction of noise into the variable-length  
7 encoding of the block of pixels, the at least one (run, level) pair is modified to produce a  
8 modified series of (run, level) pairs from the original series of (run, level) pairs. The  
9 method further includes variable-length encoding the modified series of (run, level) pairs.

10 In accordance with yet another aspect, the invention provides a method of  
11 producing MPEG encoded video from an original series of MPEG-compliant (run, level)  
12 pairs. The method includes inspecting the (run, level) pairs in the original series of (run,  
13 level) pairs to determine whether or not modification of at least one (run, level) pair in  
14 the original series of (run, level) pairs would produce a desirable decrease in a number of  
15 bits in the MPEG encoded video despite introduction of noise into the MPEG encoded  
16 video. Upon determining that modification of at least one (run, level) pair in the original  
17 series of (run, level) pairs would produce a desirable decrease in the number of bits in the  
18 MPEG encoded video despite introduction of noise into the MPEG encoded video, the at  
19 least one (run, level) pair is replaced with a sequence of a first (run, level) pair and a  
20 second (run, level) pair to produce a modified series of (run, level) pairs from the original  
21 series of (run, level) pairs. The at least one original (run, level) pair has a non-zero run  
22 length of M and a level value of N, the first (run, level) pair has run length of M-1 and a  
23 level magnitude of one, and the second (run, level) pair has a run length of zero and a

1 level value of N. The method further includes variable-length encoding the modified  
2 series of (run, level) pairs to produce the MPEG encoded video.

3 In accordance with still another aspect, the invention provides a method of  
4 decoding MPEG encoded video that includes noise introduced during the encoding  
5 process by insertion of at least one (run, level) pair having a level magnitude of one. The  
6 method includes decoding a series of (run, level) pairs from the MPEG encoded video,  
7 and inspecting the series of (run, level) pairs to find the at least one (run, level) pair  
8 having a level magnitude of one. The method further includes determining that the at  
9 least one (run, level) pair having a level magnitude of one is likely to represent noise  
10 introduced during the encoding process, and therefore rejecting the at least one (run,  
11 level) pair having a level magnitude of one in order to reduce noise.

12 In accordance with yet still another aspect, the invention provides a digital  
13 computer for producing MPEG encoded video from an original series of MPEG-  
14 compliant (run, level) pairs, the digital computer includes at least one processor  
15 programmed for inspecting the (run, level) pairs in the original series of (run, level) pairs  
16 to determine whether or not modification of at least one (run, level) pair in the original  
17 series of (run, level) pairs would produce a desirable decrease in a number of bits in the  
18 MPEG encoded video despite introduction of noise into the MPEG encoded video, and  
19 upon determining that modification of at least one (run, level) pair in the original series of  
20 (run, level) pairs would produce a desirable decrease in the number of bits in the MPEG  
21 encoded video despite introduction of noise into the MPEG encoded video, replacing the  
22 at least one (run, level) pair with a sequence of a first (run, level) pair and a second (run,  
23 level) pair to produce a modified series of (run, level) pairs from the original series of

1 (run, level) pairs. The at least one original (run, level) pair has a non-zero run length of  
2 M and a level value of N, the first (run, level) pair has a run length of M-1 and a level  
3 magnitude of one, and the second (run, level) pair has a run length of zero and a level  
4 value of N. The processor is further programmed for variable-length encoding the  
5 modified series of (run, level) pairs to produce the MPEG encoded video.

In accordance with a final aspect, the invention provides a decoder for decoding  
6  
MPEG encoded video that includes noise introduced during the encoding process by  
7  
insertion of at least one (run, level) pair having a level magnitude of one. The decoder  
8  
includes at least one processor programmed for decoding a series of (run, level) pairs  
9  
from the MPEG encoded video, inspecting the (run, level) pairs to find the at least one  
10  
(run, level) pair having a level magnitude of one, determining that the at least one (run,  
11  
level) pair having a level magnitude of one is likely to represent noise introduced during  
12  
the encoding process, and therefore rejecting the at least one (run, level) pair having a  
13  
level magnitude of one in order to reduce noise.  
14

## BRIEF DESCRIPTION OF THE DRAWINGS

17 Other objects and advantages of the invention will become apparent upon reading  
18 the following detailed description with reference to the accompanying drawings, in  
19 which:

20 FIG. 1 is a block diagram of a data network including a video file server  
21 implementing various aspects of the present invention;

22 FIG. 2 is a flowchart of a procedure executed by a stream server computer in the  
23 video file server of FIG. 1 to service client requests;

- 1 FIG. 3 is a flowchart of a procedure for splicing MPEG clips;
- 2 FIG. 4 is a flowchart of a procedure for seamless video splicing of MPEG clips;
- 3 FIG. 5 is a more detailed flowchart of the procedure for seamless video splicing
- 4 of MPEG clips;
- 5 FIG. 6 is a continuation of the flowchart begun in FIG. 5;
- 6 FIG. 7 is a timing diagram showing a timing relationship between video
- 7 presentation units (VPUs) and associated audio presentation units (APUs) in an original
- 8 MPEG-2 coded data stream;
- 9 FIG. 8 is a timing diagram showing a timing relationship between video
- 10 presentation units (VPUs) and associated audio presentation units (APUs) for a fast-
- 11 forward trick-mode stream;
- 12 FIG. 9 is a flowchart of a procedure for selection and alignment of audio
- 13 presentation units (APUs) in the fast-forward trick-mode stream;
- 14 FIG. 10 is a flowchart of a procedure for producing a trick-mode MPEG-2
- 15 transport stream from a regular MPEG-2 transport stream (TS);
- 16 FIG. 11 is a diagram illustrating relationships between the MPEG discrete cosine
- 17 transform (DCT) coefficients, spatial frequency, and the typical zig-zag scan order;
- 18 FIG. 12 is a diagram illustrating a relationship between an MPEG-2 coded bit
- 19 stream and a reduced-quality MPEG-2 coded bit stream resulting from truncation of high-
- 20 order DCT coefficients;
- 21 FIG. 13 is a flowchart of a procedure for scaling MPEG-2 coded video using a
- 22 variety of techniques;

1 FIG. 14 is a flowchart of a procedure for signal-to-noise ratio scaling MPEG-2  
2 coded video using a frequency-domain low-pass truncation (FDSNR\_LP) technique;

3 FIG. 15 is a flowchart of a procedure for signal-to-noise ratio scaling MPEG-2  
4 coded video using a frequency-domain largest-magnitude coefficient selection  
5 (FDSNR\_LM) technique;

6 FIG. 16 is a flowchart of a procedure that selects one of a number of techniques  
7 for finding a certain number "k" of largest values out of a set of "n" values;

8 FIG. 17 is a flowchart of a procedure for finding a certain number "k" of largest  
9 values from a set of "n" values, which is used in the procedure of FIG. 16 for the case of  
10  $k < \frac{1}{2} n$ ;

11 FIG. 18 is a diagram of a hash table and associated hash lists;

12 FIG. 19 is a flowchart of a procedure for finding a certain number "k" of values  
13 that are not less than the smallest of the "k" largest values in a set of "n" values beyond a  
14 certain amount.

15 FIG. 20 is a flowchart of modification of the procedure of FIG. 15 in order to  
16 possibly eliminate escape sequences in the (run, level) coding of the largest magnitude  
17 coefficients;

18 FIG. 21 is a flowchart of a subroutine called in the flowchart of FIG. 20 in order  
19 to possibly eliminate an escape sequence;

20 FIG. 22 is a first portion of a flowchart of a procedure for scaling an MPEG-2  
21 coded video data stream using the modified procedure of FIG. 20 while adjusting the  
22 parameter "k" to achieve a desired bit rate, and adjusting a quantization scaling factor  
23 (QSF) to achieve a desired frequency of occurrence of escape sequences;

1 FIG. 23 is a second portion of the flowchart begun in FIG. 22;

2 FIG. 24 is a simplified block diagram of a volume containing a main file, a  
3 corresponding fast forward file for trick mode operation, and a corresponding fast reverse  
4 file for trick mode operation;

5 FIG. 25 is a more detailed block diagram of the volume introduced in FIG. 24;

6 FIG. 26A is a diagram showing video file access during a sequence of video  
7 operations including transitions between the main file, the related fast forward file, and  
8 the related fast reverse file;

9 FIG. 26B shows a script of a video command sequence producing the sequence of  
10 video play shown in FIG. 26A;

11 FIG. 27 is a table of read and write access operations upon the volume of FIG. 24  
12 and access modes that are used for the read and write access operations;

13 FIG. 28 is a hierarchy of video service classes associated with the fast forward file  
14 and the fast reverse file in the volume of FIG. 25;

15 FIG. 29 shows a system for modifying and combining an MPEG-2 audio-visual  
16 transport stream with an MPEG-2 closed-captioning transport stream to produce a  
17 multiplexed MPEG-2 transport stream having the same bit rate as the original MPEG-2  
18 audio-visual transport stream;

19 FIG. 30 shows a flowchart of a procedure for signal-to-noise ratio scaling MPEG-  
20 2 coded video using a frequency-domain largest magnitude indices selection  
21 (FDSNR\_LMIS) technique;

22 FIG. 31 shows a graph of the picture signal-to-noise ratio (PSNR) as a function of  
23 the number of bits used for only AC coefficients' encoding using the largest magnitude

1 coefficient selection (LMCS) and largest magnitude indices selection (LMIS) procedures  
2 for quantization scale values (qsv) of 2, 4, 6, and 8, without the insertion of pivots;

3 FIG. 32 shows a graph of the picture signal-to-noise ratio (PSNR) as a function of  
4 the number of bits used for only AC coefficients' encoding using the largest magnitude  
5 coefficient selection (LMCS) and largest magnitude indices selection (LMIS) procedures  
6 for quantization scale values (qsv) of 12, 16, 20, and 24, without the insertion of pivots;

7 FIG. 33 shows a flowchart showing the successive application of a Pivot-1  
8 technique, a Pivot-2 technique, and a Pivot 3 technique for selection or insertion of pivot  
9 indices in order to avoid escape sequences or reduce the number of bits for (run, level)  
10 encoding;

11 FIG. 34 shows a graph of the average number of escape sequences per frame and  
12 a function of the number of AC coefficients retained in each block for a quantization  
13 scale value (qsv) of four for largest magnitude coefficient selection (LMCS) for no pivot  
14 insertion and for pivot insertion by each of the Pivot-1, Pivot-2, and Pivot-3 techniques;

15 FIG. 35 shows a graph of the average number of escape sequences per frame and  
16 a function of the number of AC coefficients retained in each block for a quantization  
17 scale value (qsv) of twenty-four for largest magnitude coefficient selection (LMCS) for  
18 no pivot insertion and for pivot insertion by each of the Pivot-1, Pivot-2, and Pivot-3  
19 techniques;

20 FIG. 36 shows a series of coefficients in a scan order, in order to illustrate the run  
21 length for a non-zero AC coefficient, and the insertion of a pivot index to reduce the run  
22 length;

1 FIG. 37 shows a pivot table indicating whether or not a pivot should be inserted  
2 for a given run length and level magnitude;

3 FIG. 38 shows a first sheet of a flowchart of a specific implementation of pivot  
4 insertion for avoiding escape sequences;

5 FIG. 39 is a second sheet of the flowchart begun in FIG. 38;

6 FIG. 40 is a flowchart of a procedure for a lookup in the pivot table of FIG. 37;

7 FIG. 41 is a flow diagram showing an encoding and decoding sequence including  
8 the insertion of noise in the form of pivot indices during encoding to reduce the number  
9 of bits in (run, level) encoding, and partial removal of the noise during decoding;

10 FIG. 42 is a flowchart showing how the process of pivot insertion during  
11 encoding or transcoding may be different depending on whether or not the decoder will  
12 attempt removal of the pivots;

13 FIG. 43 is a flowchart showing the removal of pivots during decoding;

14 FIG. 44 is a flowchart showing how the decoder determines whether or not a  
15 coefficient is possibly a pivot and whether or not a coefficient that is possibly a pivot is  
16 likely to be a pivot; and

17 FIG. 45 is a flow diagram showing the use of pivot insertion with transcoding or  
18 encoding with the process of largest magnitude indices selection (LMIS) or largest  
19 magnitude coefficient selection (LMCS).

20 While the invention is susceptible to various modifications and alternative forms,  
21 specific embodiments thereof have been shown by way of example in the drawings and  
22 will be described in detail. It should be understood, however, that it is not intended to  
23 limit the form of the invention to the particular forms shown, but on the contrary, the

1 intention is to cover all modifications, equivalents, and alternatives falling within the  
2 scope of the invention as defined by the appended claims.

3

1

## DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

1

## I. Applications for Efficient Scaling of Non-Scalable MPEG-2 Video

With reference to FIG. 1, there is shown a block diagram of a data network 20 linking a number of clients 21, 22, 23 to a video file server 24 implementing various aspects of the present invention. The video file server 24 includes at least one stream server computer 25 and a data storage system 26. The stream server computer 25 has a processor 27 and a network link adapter 28 interfacing the processor to the data network 20. The processor 27 executes a data streaming program 29 in memory 30 in order to stream MPEG coded video in real-time to the clients.

13 Client requests for real-time video are placed in client play lists 31 in order to  
14 schedule in advance video file server resources for the real-time streaming of the MPEG  
15 coded video. The play lists 31 specify a sequence of video clips, which are segments of  
16 MPEG-2 files 32, 33 in data storage 34 of the data storage system 26. The stream server  
17 processor 27 accesses a client play list in advance of the time to begin streaming MPEG  
18 coded video from a clip, and sends a video prefetch command to a storage controller 35  
19 in the data storage system 26. The storage controller responds to the video prefetch  
20 command by accessing the clip in the data storage 34 to transfer a segment of the clip to  
21 cache memory 36. When the video data of the segment needs to be sent to the client, the  
22 stream server processor 27 requests the data from the storage controller 35, and the  
23 storage controller immediately provides the video data from the cache memory 36.

1 Further details regarding a preferred construction and programming of the video file  
2 server 24 are disclosed in Duso et al., U.S. Patent 5,892,915 issued Apr. 6, 1999, entitled  
3 "System Having Client Sending Edit Commands to Server During Transmission Of  
4 Continuous Media From One Clip in Play List for Editing the Play List," incorporated  
5 herein by reference.

6 In accordance with an aspect of the invention, the stream server computer 25  
7 executes an MPEG scaling program 38 to produce reduced-quality MPEG coded video  
8 from nonscalable MPEG-2 coded video by truncating discrete cosine transform (DCT)  
9 AC coefficients from the coded blocks in the MPEG-2 coded video data. The reduced-  
10 quality MPEG coded video can be produced during ingestion of an MPEG-2 file 32 from  
11 the network 20, and stored in one or more associated files 37. Alternatively, the reduced-  
12 quality MPEG coded video in the files 37 could be produced as a background task from  
13 the MPEG-2 file 32. Reduced-quality MPEG coded video could also be produced in  
14 real-time from an MPEG-2 file 33 during streaming of the reduced-quality MPEG coded  
15 video from the stream server computer 25 to the network 20. The reduced-quality MPEG  
16 coded video is useful for a variety of applications, such as browsing and review of stored  
17 MPEG-2 assets for search and play-list generation, bit stream scaling for splicing, and  
18 bit-rate adjustment via video quality alteration for services with limited resources.

19 A typical example of browsing for play-list generation involves searching stored  
20 assets in a multi-media data base for segments of a desired content to be included in the  
21 play list, and in particular selecting the beginning frame and ending frame of each  
22 segment to be included. Such editing occurs often in the broadcast environment for  
23 inserting commercials and news clips into pre-recorded television programming, and for

1 editing movies for content and time compression. The decoding technique of the present  
2 invention permits a PC workstation 23 to perform the decoding and display in real-time  
3 by execution of a software program. An operator can view the video content in a display  
4 window 39 in a fast-forward or fast-reverse mode, stop at and resume from freeze frames  
5 that are valid “in points” and “out points” for seamless splicing, and select an in-point  
6 and out-point for a next segment to be included in the play list. The stream server  
7 computer 25 could also include a seamless splicing program 40 providing seamless  
8 transitions between video segments that are contiguous in a play list and are from  
9 different video clips.

10 For seamless splicing, it is often necessary to reduce the bitrate for one or more  
11 frames at the end of a first segment prior to splicing to a second segment. In this case the  
12 bitrate must be reduced to avoid buffer overflow as a result of displaying the original  
13 frames at the end of the first segment. One method of reducing the bitrate is to insert a  
14 freeze frame at the end of the first segment, but this has the disadvantage of introducing  
15 distortion in the temporal presentation of the frames and precluding frame accuracy. A  
16 less disruptive method is to use the present invention for reducing the bitrate for a lower-  
17 quality presentation of one or more frames at the end of the first segment.

18 The present invention can also reduce the transmission bit rate and storage  
19 requirements for MPEG-2 applications by altering the video quality. For example,  
20 different clients may present different bandwidth access requests for video from  
21 nonscalable MPEG-2 files 32, 33 in the video file server. Also, temporary network  
22 congestion may limit the bandwidth available to satisfy a request for real-time streaming

1 of video data. In each case, the present invention can alter the video quality to meet the  
2 desired or available bandwidth to satisfy the request.

3 With reference to FIG. 2, there is shown a flowchart of a procedure executed by a  
4 stream server computer in the video file server of FIG. 1 to service client requests. In a  
5 first step 50, execution branches to step 51 when a client request is not a request for real-  
6 time streaming. If the request is a request to input a new MPEG-2 file, then execution  
7 branches to step 52 to input the new MPEG-2 file and to create a reduced-quality version  
8 of the MPEG-2 file as available resources permit. If the request is not a request to input a  
9 new MPEG-2 file, then execution continues from step 51 to step 53. In step 53,  
10 execution branches to step 54 if the request is for play list editing. In step 54, the client  
11 may browse through the reduced-quality MPEG file to select in-points and out-points of  
12 clips to be spliced.

13 In step 50, when the request is for real-time streaming, then execution branches to  
14 step 55. In step 55, if there is network congestion so that there is insufficient bandwidth  
15 to transmit a stream of original-quality MPEG-2 coded video, then execution branches to  
16 step 56 to stream compressed video from the reduced-quality MPEG file. If no reduced-  
17 quality MPEG file is available for the desired clip, then the reduced-quality MPEG coded  
18 video to be streamed is produced in real-time from the original-quality MPEG-2 coded  
19 video. There are also applications, such as the display of spatially down-sampled video  
20 in a small display window (39 in FIG. 1), for which the client may request reduced-  
21 quality MPEG coded video. In this case, in the absence of network congestion, execution  
22 will continue from step 55 to step 57, and branch from step 57 to step 56 for streaming of  
23 reduced-quality MPEG coded video to the client.

1 Reduced-quality MPEG coded video is also useful for "trick-mode" operation.  
2 Trick-mode refers to fast forward or fast reverse display of video, in a fashion analogous  
3 to the fast forward and fast reverse playback functions of a video cassette recorder  
4 (VCR). The problem with trick-mode operation is that the transmission rate of the  
5 MPEG stream cannot simply be increased because the transmission bandwidth would be  
6 excessive and a conventional MPEG-2 decoder will not be able to handle the increased  
7 data rate or even if the decoder would have been able to support the increased data rate,  
8 such a change in the original operating conditions is not allowable. For this reason, in  
9 trick-mode, neither the original display rate of 29.97 frames per second (for NTSC or 25  
10 frames per second for PAL) nor the original transport stream (TS) multiplex rate should  
11 change. Nor is it possible to simply decimate frames since only the I frames are  
12 independently coded, and the P frames and B frames need the content of certain other  
13 frames for proper decoding. The I frames typically occur once for every 15 frames.  
14 Assuming that this convention is followed in the encoding process, it would be possible  
15 to preserve and play each I frame from each and every group of pictures (GOP), resulting  
16 in a 15 times slower temporal sampling rate, or a 1 to 15 speeding up of motion if the I  
17 frames only are played back at the nominal NTSC rate of approximately 30 frames per  
18 second. Consequently, the content of a 60 minutes duration clip will be covered in 4  
19 minutes. Unfortunately the average information content per frame for the I frames is  
20 more than the average information content of I, P and B frames. Therefore, the trick-  
21 mode cannot be implemented simply by transmitting only the I frames for a speed-up by  
22 a factor of 15, because this would need an increase in the TS multiplex rate over the  
23 nominal rate.

In particular, in a sample analysis the average information content of an I frame has been measured to be about 56374.6 bytes. If the I frames only are transmitted at the standard NTSC rate, then the bit transmission rate would be: 8(bits per byte) \* 56,374.6(bytes per frame) \* 29.97(frames per sec.) or about 13,516,374.1 bits per second only for the video stream, which is significantly above - almost 3.38 times - the original rate of 4 megabits per second used in this test. This calculation, being based on an average quantity, is ignoring the indispensable need for an actually higher transport rate to provide some safety margin to handle short-term-sustained large size I and/or P frame chains (bursts) which practically always happen. Clearly, some form of modification in the trick-mode operation definition is required to handle this problem and pull the bit-rate requirement down to the nominal 4 megabits per second.

Two degrees of freedom are available to achieve such a reduction in the required bit-rate for trick-mode operation. The first is I frame compression quality and the second is a motion speed-up ratio. With respect to compression quality, it is well known that human observers' perception of image detail degrades with increasing motion speed of objects in the scene. Based on this fact, the type of D pictures were introduced in MPEG-1 video syntax for fast visible (forward or reverse) search purposes. (See ISO/IEC 11172-2: 1993 Information Technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s - Part 2: Video, Annex D.6.6. Coding D-Pictures, p.102). D pictures make use of only the DC coefficients in intra coding to produce very low quality (in terms of SNR) reproductions of desired frames which were judged to be of adequate quality in fast search mode.

In order to provide support for enhanced quality trick-mode operation, the quality

1 of the original I frames can be reduced by the preservation of just a sufficient number of  
2 AC DCT coefficients to meet the bit-rate limitation. Based on experiments with two  
3 standard video test sequences (one encoded at 15 Mbits/sec. and the other at 24  
4 Mbits/sec. and both with I frames only), it is observed that the bandwidth for I frames can  
5 be scaled to one half by keeping about 9 lowest order AC coefficients and eliminating the  
6 rest. This scheme provides good quality even at the full spatial and temporal resolution,  
7 much better than D pictures.

8       The inherent speed-up ratio lower bound imposed by the GOP structure can be  
9 relaxed and further lowered by freeze (P) frame substitution in between genuine (SNR  
10 scaled or non-scaled) I frames. The maximum number of freeze frames that can be  
11 inserted before visually disturbing motion jerkiness occurs, is very likely to depend  
12 heavily on the original GOP structure (equivalently the separation between I frames of  
13 the original sequence) and the original amount of motion in the clip. However, 1, 2 or 3  
14 freeze frame substitutions in between genuine I frames present reasonable choices which  
15 will yield speed-up ratios of 1 to 7.5, 1 to 5 and 1 to 3.75 respectively instead of the 1 to  
16 15 speed-up ratio provided by the genuine I frames only implementation. (These ratios  
17 are computed by a first-order approximation that neglects a slight increase in bandwidth  
18 required by the consecutive freeze frames, which are inserted in between genuine I  
19 frames and can typically be made very small in size in comparison to the average size of  
20 a genuine I frame. Therefore, the insertion of 1, 2, 3 freeze frames will result in  
21 bandwidth reductions of 2 to 1, 3 to 1 and 4 to 1 respectively. The accuracy of this  
22 approximation degrades as more consecutive freeze frames and/or SNR scaling is  
23 employed.) An easy way to see the validity of these approximate figures is to note for

example that in the case of 1 freeze frame insertion, the total presentation time of the trick-mode clip for an originally 60 minutes duration asset will increase from 4 minutes to 8 minutes. Since due to the underlying assumption of the first-order approximation stated above, the same amount of data (I frames only) will be transmitted in this doubled time interval, the bandwidth requirement will be halved. The final choice for trick-mode implementation should reflect a balanced trade-off along these two degrees of freedom.

For example, SNR scaling of I frames down to 9 AC coefficients can be used along with single freeze frame insertion between I frames. These two choices, both of which are individually capable of providing a 2 to 1 bandwidth reduction as discussed before, will yield a combined 4 to 1 bandwidth reduction which will comfortably bring the non-scaled I frame-only bit-rate of 13516374.1 bits/sec. down to below the 4 Mbits/sec. quota. If the visual quality provided by 9 AC coefficients is not considered adequate, then SNR scaling could be tuned to keep more AC coefficients at the expense of a smaller bandwidth reduction. This, however, could be compensated consequently by increasing the number of freeze frames to be used in between I frames. Coarser quantization (and therefore poorer visual quality) can be tolerated at high trick-mode speeds and better visual quality should be retained at lower trick-mode speeds.

With reference to FIG. 2, if the client has requested trick-mode operation, execution branches from step 58 to step 59. In step 59, execution branches to step 60 for a low value of speed-up. In step 60, the trick-mode stream is produced by streaming original-quality I frames and inserting three freeze frames per I frame, to yield a speed-up factor of  $15/4 = 3.75$  based on an original MPEG-2 coded stream having one I frame for every 15 frames. For a higher speed-up factor, execution branches from step 59 to step

1       61. In step 61, either one or two freeze frames are selected per I frame to provide a  
2 speed-up factor of  $15/2=7.5$ , or  $15/3 = 5$  respectively. Then in step 62 the trick-mode  
3 stream is produced by streaming reduced-quality I frames and inserting the selected  
4 number of freeze frames between the reduced-quality I frames. If a trick-mode operation  
5 is not requested in step 58, then execution continues from step 58 to step 63. In step 63,  
6 the stream server computer streams original-quality MPEG-2 coded data to the client.  
7 Further details regarding trick-mode operation are described below with reference to  
8 FIGs. 7 to 10.

9

10           II. MPEG Splicing

11           FIGs. 3 to 6 show further details regarding use of the present invention for MPEG  
12 splicing. In particular, reduced-quality frames are substituted for the freeze frames used  
13 in the seamless splicing procedure found in the common disclosure of Peter Bixby et al.,  
14 U.S. application Ser. 09/539,747 filed March 31, 2000; Daniel Gardere et al., U.S.  
15 application Ser. 09/540,347 filed March 31, 2000; and John Forecast et al. U.S.  
16 application Ser. 09/540,306 filed March 31, 2000; which are all incorporated by reference  
17 herein. The common disclosure in these U.S. applications considered pertinent to the  
18 present invention is included in the written description below with reference to FIGs. 3 to  
19 6 in the present application (which correspond to FIGs. 19, 22, 23, and 24 in each of the  
20 cited U.S. applications).

21           FIG. 3 shows a basic procedure for MPEG splicing. In the first step 121, the  
22 splicing procedure receives an indication of a desired end frame of the first clip and a  
23 desired start frame of the second clip. Next, in step 122, the splicing procedure finds the

1        closest I frame preceding the desired start frame to be the In Point for splicing. In step  
2        123, the splicing procedure adjusts content of the first clip near the end frame of the first  
3        clip and adjusts content of the second clip near the In Point in order to reduce  
4        presentation discontinuity (due to decoder buffer underflow) and also to prevent decoder  
5        buffer overflow when decoding the spliced MPEG stream. Finally, in step 124, the  
6        concatenation of the first clip up to about the Out Point and the second clip subsequent to  
7        about the In Point is re-formatted, including re-stamping of the presentation time stamps  
8        (PTS), decoding time stamps (DTS), and program clock reference (PCR) values for the  
9        audio and video streams in the second clip.

10           Considering now video splicing, the splicing procedure should ensure the absence  
11          of objectionable video artifacts, preserve the duration of the spliced stream, and if  
12          possible, keep all of the desired frames in the spliced stream. The duration of the spliced  
13          stream should be preserved in order to prevent any time drift in the scheduled play-list.  
14          In some cases, it is not possible to keep all of the original video frames due to buffer  
15          problems.

16           Management of the video buffer is an important consideration in ensuring the  
17          absence of objectionable video artifacts. In a constant bit rate (CBR) and uniform picture  
18          quality sequence, subsequent pictures typically have coded representations of drastically  
19          different sizes. The encoder must manage the decoder's buffer within several constraints.  
20          The buffer should be assumed to have a certain size defined in the MPEG-2 standard.  
21          The decoder buffer should neither overflow nor underflow. Furthermore, the decoder  
22          cannot decode a picture before it receives it in full (i.e. completely). Moreover, the  
23          decoder should not be made to "wait" for the next picture to decode; this means that

1 every 40 ms in PAL and 1/29.97 second in NTSC, the decoder must have access to a full  
2 picture ready to be decoded.

3 The MPEG encoder manages the video decoder buffer through decode time  
4 stamps (DTS), presentation time stamps (PTS), and program clock reference (PCR)  
5 values. When splicing the end of a first clip to the beginning of a second clip, there will  
6 be a problem of video buffer management if a duration of time  $DTS_{L1}-T_e$  is different from  
7 a duration of time  $DTS_{F2}-PCR_{e2}$  minus one video frame (presentation) interval, where  
8  $DTS_{L1}$  is the DTS at the end of the first clip and indicates the time at which the video  
9 decoder buffer is emptied of video data from the first clip,  $T_e$  is the time at which the last  
10 video frame's data is finished being loaded into the video decoder buffer,  $DTS_{F2}$  is the  
11 DTS of the first frame of the second clip, and  $PCR_{e2}$  is the PCR of the second clip  
12 extrapolated from the value of the most recent received genuine PCR record, to the first  
13 byte of the picture header sync word of the first video frame in the clip to start. The  
14 extrapolation adjusts this most recently received genuine PCR record value by the  
15 quotient of the displacement in data bits of the clip from the position where it appears in  
16 the second clip to the position at which video data of the first frame of the second clip  
17 begins, divided by the data transmission bit rate for transmission of the clip to the  
18 decoder. Because the time  $PCR_{e2}$  must immediately follow  $T_e$ , there will be a gap in the  
19 decoding and presentation of video frames if  $DTS_{F2}-PCR_{e2}$  is substantially greater than  
20  $DTS_{L1}-T_e$  plus one video frame interval. In this case, the buffer will not be properly full  
21 to begin decoding of the second clip one video frame interval after the last frame of the  
22 first clip has been decoded. Consequently, either the second clip will be prematurely  
23 started to be decoded or the decoder will be forced to repeat a frame one or more times

1 after the end of the display of the last frame from the first clip to provide the required  
2 delay for the second clip's buffer build-up. In the case of a premature start for decoding  
3 the second clip, a video buffer underflow risk is generated. On the other hand, in case of  
4 repeated frames, the desired frame accuracy for scheduled play-lists is lost besides the  
5 fact that neither a guaranteed safe buffer management can be achieved through this  
6 procedure.

7 If  $DTS_{F2}-PCR_{e2}$  is substantially less than  $DTS_{L1}-T_e$  plus one video frame interval,  
8 then the decoder will not be able to decode the first frame of the second clip at the  
9 specified time  $DTS_{F2}$  because either the last frame of the first clip will not yet have been  
10 removed from the video buffer or the last frame of the first clip has already been moved  
11 but the frame interval duration required before decoding the next frame has not elapsed  
12 yet. In this case a video buffer overflow risk is generated. Video buffer overflow may  
13 present a problem not only at the beginning of the second clip, but also at a subsequent  
14 location of the second clip. If the second clip is encoded by an MPEG-2 compliant  
15 encoder, then video buffer underflow or buffer overflow will not occur at any time during  
16 the decoding of the clip. However, this guarantee is no longer valid if the  $DTS_{F2}-PCR_{e2}$   
17 relationship at the beginning of the second clip is altered. Consequently, to avoid buffer  
18 problems, the buffer occupancy at the end of the first clip must be modified in some  
19 fashion. This problem is inevitable when splicing between clips having significantly  
20 different ending and starting buffer levels. This is why the Society of Motion Picture and  
21 Television Engineers (SMPTE) has defined some splice types corresponding to well-  
22 defined buffer levels. (See SMPTE Standard 312M, entitled "Splice Points for MPEG-2  
23 Transport Streams," SMPTE Journal, Nov. 1998.) In order to seamlessly splice the first

1 clip to the second clip, the content of the first clip (towards its end) is modified so that  
2 PCR<sub>e2</sub> can immediately follow T<sub>e</sub> (by one byte transmission time) and DTS<sub>F2</sub> can just  
3 follow DTS<sub>L1</sub> (by one video frame presentation interval).

4 FIG. 4 shows a flow chart of a seamless video splicing procedure that attains the  
5 desired condition just described above. In a first step 141, the first DTS of the second  
6 clip is anchored at one frame interval later than the last DTS of the first clip in order to  
7 prevent a video decoding discontinuity. Then, in step 142, the procedure branches  
8 depending on whether the PCR extrapolated to the beginning frame of the second clip  
9 falls just after the ending time of the first clip. If so, then the splice will be seamless with  
10 respect to the original video content. Otherwise, the procedure branches to step 143. In  
11 step 143, the content of the first clip is adjusted so that the PCR extrapolated to the  
12 beginning frame of the second clip falls just after the ending time of the first clip.  
13 Therefore the desired conditions for seamless video splicing are achieved.

14 With reference to FIG. 5, there is shown a more detailed flow chart of a seamless  
15 video splicing procedure. In a first step 151, the procedure inspects the content of the  
16 first clip to determine the last DTS/PTS of the first clip. This last DTS/PTS of the first  
17 clip is designated DTS<sub>L1</sub>. Next, in step 152, the procedure inspects the content of the first  
18 clip to determine the time of arrival (T<sub>e</sub>) of the last byte of the first clip. In step 153, the  
19 procedure adds one frame interval to DTS<sub>L1</sub> to find the desired first DTS location for the  
20 second clip. The sum, designated DTS<sub>F1</sub>, is equal to DTS<sub>L1</sub> +1/FR, where FR is the video  
21 frame rate. In step 154, while keeping the DTS<sub>F2</sub>-PCR<sub>e2</sub> relationship unaltered for the  
22 second clip, the procedure finds the time instant, designated T<sub>S</sub>, at which the first byte of

1 the second clip should arrive at the decoder buffer. This is done by calculating  
2  $T_{START}=DTS_{F2}-PCR_{e2}$ , and  $T_S=DTS_{F1}-T_{START}$ .

3 Continuing in FIG. 6, in step 155, execution branches depending on whether  $T_S$  is  
4 equal to  $T_e$  plus 8 divided by the bit rate. If not, then the clips to be spliced need  
5 modification before concatenation, and execution branches to step 156. In step 156,  
6 execution branches depending on whether  $T_S$  is less than  $T_e$  plus 8 divided by the bit rate.  
7 If not, then there is an undesired gap in between the clips to be spliced, and execution  
8 branches to step 157. In step 157, null packets are inserted into the clips to be spliced to  
9 compensate for the gap. The gap to be compensated has a number of bytes, designated  
10  $G_r$ , equal to  $(T_S-T_e)(BIT\ RATE)/8$  minus one. If in step 156,  $T_S$  is less than  $T_e$  plus 8  
11 divided by the bit rate, then execution continues from step 156 to step 158 to open up a  
12 certain amount of space in the first clip to achieve  $T_S=T_e+8/(BIT\ RATE)$ . The number of  
13 bytes to drop is one plus  $(T_e-T_S)(BIT\ RATE)/8$ . If possible, the bytes are dropped by  
14 removing null packets. Otherwise, one or more frames at the end of the first clip are  
15 replaced with corresponding reduced-quality frames, which have fewer bytes than the  
16 original-quality frames at the end of the first clip.

17 If in step 155  $T_S$  is found to be equal to  $T_e$  plus 8 divided by the bit rate, then  
18 execution continues to step 159. Execution also continues to step 159 from steps 157 and  
19 158. In step 159, the transport streams from the two clips are concatenated. Finally, in  
20 step 160, a subroutine is called to compute a video time stamp offset, designated as  
21  $V_{OFFSET}$ . This subroutine finds the DTS of the last video frame (in decode order) of the  
22 first clip. This DTS of the last video frame of the first clip is denoted  $DTS_{VL1}$ . Then the  
23 subroutine finds the original DTS of the first frame to be decoded in the second clip.

1 This DTS of the first frame to be decoded in the second clip is denoted DTS<sub>VF2</sub>. Finally,  
2 the subroutine computes the video time stamp offset V<sub>OFFSET</sub> as DTS<sub>VL1</sub>-DTS<sub>VF2</sub> plus one  
3 video frame duration.

4

5       III. Trick Mode Operation

6 FIGs. 7 to 10 show further details regarding trick-mode operation. FIG. 7 shows  
7 a timing relationship between video presentation units (VPUs) and associated audio  
8 presentation units (APUs) in an original MPEG-2 coded data stream, and FIG. 8 shows  
9 similar timing for the fast-forward trick-mode stream produced from the original data  
10 stream of FIG. 7. (The fast-forward trick-mode stream is an example of a trick-mode  
11 stream that could be produced in step 60 of FIG. 2.) The original data stream has  
12 successive video presentation units for video frames of type I, B, B, P, B respectively.  
13 The trick-mode stream has successive video presentation units for video frames of types  
14 I, F, F, I, F where “F” denotes a freeze P (or possibly B) frame. Each I frame and  
15 immediately following F frames produce the same video presentation units as a  
16 respective I frame in the original data stream of FIG. 7, and in this example, one in every  
17 15 frames in the original data stream is an I frame. Each freeze frame is coded, for  
18 example, as a P frame repeating the previous I frame or the previous P-type freeze-frame  
19 (in display order). In each freeze frame, the frame is coded as a series of maximum-size  
20 slices of macroblocks, with an initial command in each slice indicating that the first  
21 macroblock is an exact copy of the corresponding macroblock in the previous frame  
22 (achieved by predictive encoding with a zero valued forward motion compensation vector  
23 and no encoded prediction error), and two consequent commands indicating that the

1 following macroblocks in the slice until and including the last macroblock of the slice are  
2 all coded in the same way as the first macroblock.

3 For trick-mode operation, there is also a problem of how to select audio  
4 presentation units (APU) to accompany the video presentation units that are preserved in  
5 the trick-mode stream. Because the video presentation units (VPU) have a duration of  
6 ( $1/29.97$ ) sec. or about 33.37 msec. and the audio presentation units (APU) typically have  
7 a duration of 24 msec., there is neither a one-to-one correspondence nor alignment  
8 between VPUs and APUs. In a preferred implementation, the audio content of a trick-  
9 mode clip is constructed as follows. Given the total presentation duration ( $1/29.97$ ) sec.  
10 or about 33.37 msec. for a single video frame, it is clear that always at least one and at  
11 most two 24 msec. long audio presentation units (APU) will start being presented during  
12 the end-to-end presentation interval of each video frame. This statement refers to the  
13 original clip and does not consider any audio presentation unit whose presentation is  
14 possibly continuing as the video frame under consideration is just put on the display. The  
15 first of the above mentioned possibly two audio presentation units will be referred to as  
16 the aligned audio presentation unit with respect to the video frame under consideration.  
17 For example, in FIG. 8, the  $APU_j$  is the aligned audio presentation unit with respect to the  
18  $VPU_i$ . Now, when the I frames are extracted and possibly SNR scaled and possibly  
19 further interleaved with a number of freeze P frames in between them to produce the  
20 trick-mode video packetized elementary stream (PES), the associated trick-mode audio  
21 stream is constructed as follows. For each I type video frame presentation interval (and  
22 for that matter also for freeze P type video frames) in this trick-mode clip, the above  
23 stated fact of at least one (and at most two) audio presentation unit being started, holds.

1 Then for each I frame presentation interval in the trick-mode clip, once any possibly  
2 previously started and continuing audio presentation unit ends, insert its aligned audio  
3 presentation unit (from the original clip) and continue inserting APUs from the original  
4 clip subsequent to the aligned one until covering the rest of the I frame presentation  
5 interval and also any possibly following freeze P frame presentation intervals until  
6 crossing into and overlapping (or less likely aligning) with the next I frame's presentation  
7 interval. In FIG. 8, for example, the audio presentation units APU<sub>j</sub>, APU<sub>j+1</sub>, APU<sub>j+2</sub>, and  
8 APU<sub>j+3</sub> are inserted, until crossing into and overlapping with the next I frame VPU<sub>i+15</sub>.  
9 Following APU<sub>j+3</sub> is inserted APU<sub>k</sub>, which designates the APU aligned with VPU<sub>i+15</sub> in  
10 the original stream. Clearly, the final alignment of (the aligned and consequent) audio  
11 presentation units with respect to their associated I frames will be slightly different in the  
12 trick-mode clip as compared to the original clip. However, considering how the trick-  
13 mode audio component will sound like, this poses no problem at all.

14 FIG. 9 is a flowchart of a procedure for producing the desired sequencing of audio  
15 presentation units (APUs) in the fast-forward trick-mode stream. This procedure scans  
16 the audio elementary stream in the original MPEG-2 stream to determine the sequence of  
17 APUs in the original stream and their presentation-time alignment with the I frames in the  
18 video elementary stream of the original MPEG-2 transport stream, while selecting APUs  
19 to include in the trick-mode stream. In a first step 171, execution proceeds once the end  
20 of the current APU is reached. If the end of the current APU has not entered a new VPU  
21 (*i.e.*, the beginning of the current APU is within the presentation time of one VPU and the  
22 end of the current APU is within the presentation time of the same VPU), or if it has  
23 entered a new VPU (*i.e.*, the beginning of the current APU is within the presentation time

1 of one VPU and the end of the current APU is within the presentation time of a new  
2 (next) VPU but the new VPU is not an I frame, then execution branches to step 174. In  
3 step 174, an APU pointer is incremented, and in step 175 execution proceeds into this  
4 next APU. If in step 173 the end of the current APU extends into an I frame, then in step  
5 176 the APU pointer is advanced to point to the first APU beginning within the duration  
6 of the VPU of the I frame in the original MPEG-2 stream.

7 FIG. 10 is a flowchart of a procedure for producing a trick-mode stream from an  
8 MPEG-2 transport stream (TS). In a first step 181, the MPEG-2 TS is inputted. In step  
9 182, the video elementary stream (VES) is extracted from the TS. In step 183, a  
10 concurrent task extracts the audio elementary stream (AES) from the TS. In step 184, I  
11 frames are extracted from the VES and valid packetized elementary stream (PES) packets  
12 are formed encapsulating the I frames. In step 185, the I frames are SNR scaled, for the  
13 high speed cases of the trick-mode. In step 186, P-type freeze frames are inserted into the  
14 stream of SNR scaled I frames (in between the scaled I frames), and valid PES packets  
15 are formed for the trick-mode VES encapsulating the P-type freeze frames and the SNR  
16 scaled I frames. Concurrently, in step 187, appropriate audio access units (from the  
17 originally input MPEG-2 TS asset) are selected and concatenated based on the structure  
18 of the VES being formed for the trick-mode clip, as described above with reference to  
19 FIG. 9, and valid PES packet encapsulation is formed around these audio access units.  
20 Finally, in step 188, the trick-mode TS stream is generated by multiplexing the trick-  
21 mode VES from step 186 into a system information (SI) and audio PES carrying TS  
22 skeleton including the audio PES packets from step 187.

23

1           IV. Truncation of AC DCT Coefficients for Producing Low-Quality MPEG  
2       Coded Video

3           FIGs 11 to 19 include details of the preferred techniques for truncating AC DCT  
4       coefficients for producing low-quality MPEG-2 coded video from original-quality  
5       MPEG-2 coded video. Most of these techniques exploit the fact that in the typical  
6       (default) zig-zag scan order, the basis functions for the high-order AC DCT coefficients  
7       have an increasing frequency content. FIG 11, for example, shows a matrix of the DCT  
8       coefficients  $C_{ij}$ . The row index ( $i$ ) increases with increasing vertical spatial frequency in  
9       a corresponding 8x8 coefficient block, and the column index ( $j$ ) increases with increasing  
10      horizontal spatial frequency in the corresponding 8x8 coefficient block. The coefficient  
11       $C_{11}$  has zero frequency associated with it in both vertical and horizontal directions, and  
12      therefore it is referred to as the DC coefficient of the block. The other coefficients have  
13      non-zero spatial frequencies associated with their respective basis functions, and  
14      therefore they are referred to as AC coefficients. Each coefficient has an associated basis  
15      function  $f_{ij}(x,y)$  that is separable into  $x$  and  $y$  components such that  $f_{ij}(x,y)=f_i(y)f_j(x)$ .  
16      The  $x$  and  $y$  component functions  $f_i(y)$  and  $f_j(x)$  are shown graphically in FIG. 11 as  
17      cosine functions in order to illustrate their associated spatial frequencies. In practice, the  
18      component functions are evaluated at discrete points for the 64 pixel positions in the 8x8  
19      blocks, so that each of the DCT basis functions is an 8x8 array of real numbers. In  
20      particular, the component functions are:

21  
22       $f_i(y)=\text{SQRT}((2-\delta_{i-1})/8)(\cos((\pi/8)(y-1/2)(i-1)))$  for  $y=1, 2, 3, \dots, 8$   
23       $f_j(x)=\text{SQRT}((2-\delta_{j-1})/8)(\cos((\pi/8)(x-1/2)(j-1)))$  for  $x=1, 2, 3, \dots, 8$

1 where  $\delta_0 = 1$ , and  $\delta_p = 0$  for  $p > 0$ . The path including a number of diagonal line segments  
2 through the matrix of coefficients in FIG. 11 denotes the default zig-zag scan order  
3 typically used for MPEG-2 encoding. Listed in this order, the coefficients are  $C_{11}$ ,  $C_{12}$ ,  
4  $C_{21}$ ,  $C_{31}$ ,  $C_{22}$ ,  $C_{13}$ ,  $C_{14}$ ,  $C_{23}$ ,  $C_{32}$ ,  $C_{41}$ , ...,  $C_{86}$ ,  $C_{77}$ ,  $C_{68}$ ,  $C_{78}$ ,  $C_{87}$ ,  $C_{88}$ . The first coefficient  
5 in this zig-zag scan order is the DC coefficient  $C_{11}$  providing the lowest spatial frequency  
6 content in the 8x8 block of pixels, and the last coefficient in this zig-zag scan order is the  
7 coefficient  $C_{88}$  providing the highest spatial frequency content in the 8x8 block of pixels.

8 FIG. 12 is a diagram illustrating a relationship between an original MPEG-2  
9 coded bit stream 200 and a reduced-quality MPEG-2 coded bit stream 210 resulting from  
10 truncation of high-order DCT coefficients from the original MPEG-2 coded bit stream.  
11 Shown in the original MPEG-2 coded bit stream 200 is a portion of a video PES packet  
12 including DCT coefficients for an 8x8 pixel block. The DCT coefficients include a  
13 differentially coded DC coefficient 201, and three (run, level) events 202, 203, 204  
14 encoding three respective nonzero AC coefficients possibly along with some zero valued  
15 AC coefficients preceding the three nonzero valued ones. The DCT coefficients are  
16 ordered according to the zig-zag scan order shown in FIG. 11 (or possibly according to an  
17 alternate zig-zag scan pattern also supported by the MPEG-2 standard), and AC  
18 coefficients having zero magnitude are described in terms of total counts of consecutive  
19 zero valued coefficients lying in between two nonzero valued coefficients, in the MPEG-  
20 2 coded bit stream. An end-of-block (EOB) code 205 signals the end of the encoded  
21 DCT coefficients for the current block. The reduced-quality MPEG-2 coded bit stream  
22 210 includes a DC coefficient 201' identical to the DC coefficient 201 in the original  
23 MPEG-2 coded bit stream 200, and a (run, level) event 202' identical to the (run, level)

1 event 202 in the original MPEG-2 coded bit stream 200. Second and third (run, level)  
2 events, however, have been omitted from the reduced-quality MPEG-2 bit stream 210,  
3 because an EOB code 205' immediately follows the (run, level) event 202'. Therefore,  
4 the two nonzero high-order AC DCT coefficients encoded by the second and third (run,  
5 level) events 203, 204 have been omitted from the reduced-quality MPEG-2 bit stream  
6 210.

7 FIG. 13 is a flowchart of a procedure for scaling MPEG-2 coded video using a  
8 variety of techniques including the omission of AC DCT coefficients. The procedure  
9 operates upon an original-quality MPEG-2 coded video stream by removing AC DCT  
10 coefficients in this stream to produce a lower quality MPEG coded video stream. In a  
11 first step 221, execution branches to step 222 if the scaled MPEG coded video is to be  
12 spatially subsampled. In step 222, the procedure removes any and all DCT coefficients  
13 for spatial frequencies in excess of the Nyquist frequency for the downsampled video.  
14 For example, if the low-quality video stream will be downsampled by a factor of two in  
15 both the vertical and the horizontal directions, then the procedure removes any and all  
16 DCT coefficients having a row index (i) greater than four and any and all DCT  
17 coefficients having a column index (j) greater than four. This requires the decoding of  
18 the (run, level) coded coefficients to the extent necessary to obtain an indication of the  
19 coefficient indices. If a sufficient number of the original AC DCT coefficients are  
20 removed for a desired bandwidth reduction, then the scaling procedure is finished.  
21 Otherwise, execution branches from step 223 to step 224. Execution also continues from  
22 step 221 to step 224 if spatial downsampling is not intended.

1           In step 224, execution branches to step 225 if low-pass scaling is desired. Low-  
2       pass scaling requires the least computational resources and may produce the best results  
3       if the scaled, low-quality MPEG coded video is spatially downsampled. In step 225, the  
4       procedure retains up to a certain number of lowest-order AC DCT coefficients for each  
5       block and removes any additional DCT coefficients for each block. This is a kind of  
6       frequency domain signal-to-noise ratio scaling (FDSNR) that will be designated  
7       FDSNR\_LP. A specific example of the procedure for step 225 will be described below  
8       with reference to FIG. 14.

9           Execution continues from step 224 to step 226 if low-pass scaling is not desired.  
10          In step 226, execution branches to step 227 if largest magnitude based scaling is desired.  
11          Largest magnitude based scaling produces the least squared error or difference between  
12       the original-quality MPEG-2 coded video and the reduced-quality MPEG coded video for  
13       a given number of nonzero AC coefficients to preserve, but it requires more  
14       computational resources than the low-pass scaling of step 225. More computational  
15       resources are needed because if there are more nonzero AC coefficients than the desired  
16       number of AC coefficients for a block, then the (run, level) events must be decoded fully  
17       to obtain the coefficient magnitudes, and additional resources are required to find the  
18       largest magnitude coefficients. In step 227, the procedure retains up to a certain number  
19       of largest magnitude AC DCT coefficients for each block, and removes any and all  
20       additional AC DCT coefficients for each block. This is a kind of frequency domain  
21       signal-to-noise ratio scaling (FDSNR) that will be designated FDSNR\_LM. A specific  
22       example of the procedure for step 227 will be described below with reference to FIG. 15.

1        If in step 226 largest magnitude based scaling is not desired, then execution  
2        continues to step 228. In step 228, execution branches to step 229 to retain up to a certain  
3        number of AC DCT coefficients that differ in magnitude from up to that number of  
4        largest magnitude AC DCT coefficients by no more than a certain limit. This permits a  
5        kind of approximation to FDSNR\_LM in which an approximate search is undertaken for  
6        the largest magnitude AC DCT coefficients if there are more nonzero AC DCT  
7        coefficients than the desired number of AC DCT coefficients in a block. The  
8        approximate search can be undertaken using a coefficient magnitude classification  
9        technique such as a hashing technique, and the low-pass scaling technique can be applied  
10      to the classification level that is incapable of discriminating between the desired number  
11      of largest magnitude AC DCT coefficients. A specific example is described below with  
12      reference to FIG. 19.

13       With reference to FIG. 14, there is shown a flowchart of a procedure for scaling  
14      MPEG-2 coded video using the low-pass frequency-domain signal-to-noise (FDSNR\_LP)  
15      scaling technique. This procedure scans and selectively copies components of an input  
16      stream of original-quality MPEG-2 coded video to produce an output stream of reduced-  
17      quality MPEG-2 coded video. The procedure is successively called, and each call  
18      processes coefficient data in the input stream for one 8x8 block of pixels. No more than a  
19      selected number “k” of coded lowest order (nonzero or zero valued) AC coefficients are  
20      copied for the block where the parameter “k” can be specified for each block.

21       In a first step 241 of FIG. 14, the procedure parses and copies the stream of  
22      original-quality MPEG-2 coded data up to and including the differential DC coefficient  
23      variable-length code (VLC). Next, in step 242, a counter variable “l” is set to zero. In

1 step 243, the procedure parses the next (run, level) event VLC in the stream of original-  
2 quality MPEG-2 coded data. In step 244, if the VLC just parsed is an end-of-block  
3 (EOB) marker, execution branches to step 245 to copy the VLC to the stream of reduced-  
4 quality MPEG-2 coded video, and the procedure is finished for the current block.

5       In step 244, if the VLC just parsed is not an EOB marker, then execution  
6 continues to step 246. In step 246, a variable “r” is set equal to the run length of zeroes  
7 for the current (run, level) event, in order to compute a new counter value  $l+r+1$ . In step  
8 247, if the new counter value  $l+r+1$  is greater than the parameter “k”, then the procedure  
9 branches to step 248 to copy an EOB marker to the stream of reduced-quality MPEG  
10 coded data. After step 248, execution continues to step 249, where the procedure parses  
11 the input stream of original-quality MPEG-2 coded data until the end of the first EOB  
12 marker, and the procedure is finished for the current block.

13       In step 247, if the new counter value  $l+r+1$  is not greater than the parameter “k”,  
14 then execution continues to step 250. In step 250, execution branches to step 251 if the  
15 new counter value  $l+r+1$  is not equal to “k” (which would be the case if the new counter  
16 value is less than “k”). In step 251, the counter state  $l$  is set equal to the new counter  
17 value  $l+r+1$ . Then, in step 252, the VLC just parsed (which will be a VLC encoding a  
18 (run, level) event) is copied from the stream of original-quality MPEG-2 coded data to  
19 the stream of reduced-quality MPEG-2 coded data. After step 252, execution loops back  
20 to step 243 to continue the scanning of the stream of original-quality MPEG-2 coded  
21 data.

22       In step 250, if the new counter value  $l+r+1$  is equal to “k”, then execution  
23 branches from step 250 to step 253, to copy the VLC just parsed (which will be a VLC

1 encoding a (run, level) event from the stream of original-quality MPEG-2 coded data to  
2 the stream of reduced-quality MPEG-2 coded data. Next, in step 254, the procedure  
3 copies an EOB marker to the stream of reduced-quality MPEG-2 coded data. After step  
4 254, execution continues to step 249, where the procedure parses the input stream of  
5 original-quality MPEG-2 coded data until the end of the first EOB marker, and the  
6 procedure is finished for the current block.

7 FIG. 15 is a flowchart of a procedure for scaling MPEG-2 coded video using the  
8 largest magnitude based frequency-domain signal-to-noise ratio (FDSNR\_LM) scaling  
9 technique. This routine is successively called, and each call processes coefficient data in  
10 the input stream for one 8x8 block of pixels. No more than a specified number “k” of  
11 largest magnitude AC DCT coefficients are copied for the block, and a different number  
12 “k” can be specified for each block.

13 In a first step 261 in FIG. 15, the procedure parses and copies the input stream of  
14 original-quality MPEG-2 coded data to the output stream of lower-quality MPEG-2 data  
15 up to and including the differential DC coefficient variable-length code (VLC). Then in  
16 step 262 all (run, level) event VLCs are parsed and decoded until and including the EOB  
17 marker of the current block. The decoding produces coefficient identifiers and  
18 corresponding quantization indices representing the quantized coefficient values. In step  
19 263, the quantization indices are transformed to quantized coefficient values. In step 264,  
20 the (quantized) coefficients are sorted in descending order of their magnitudes. In step  
21 265, the first “k” coefficients of the sorted list are preserved and the last 63-k AC DCT  
22 coefficients in the sorted list are set to zero. In step 266, (run, level) event formation and  
23 entropy coding (VLC encoding) are applied to the new set of coefficient values. Finally,

1   in step 267, the VLCs resulting from step 266 are copied to the output stream until and  
2   including the EOB marker.

3           The sorting step 264 of the FDSNR\_LM procedure can consume considerable  
4   computational resources. It is important to notice that not a full sorting of the quantized  
5   AC coefficients with respect to their magnitudes but rather a search for a specified  
6   number “ $k$ ” of largest magnitude AC coefficients is all that is required. This task can be  
7   performed exactly or approximately in different ways so as to avoid the complexity  
8   associated with a conventional sorting procedure. In general, a relatively large number of  
9   the 63 AC DCT coefficients will have a quantized value of zero. Only the non-zero  
10   coefficients need be included in the sorting process. Moreover, if there are “ $n$ ” non-zero  
11   coefficients and only “ $k$ ” of them having the largest magnitudes are to be preserved in the  
12   output stream, then the sorting process may be terminated immediately after only the  
13   largest magnitude “ $k$ ” coefficients have been found, or equivalently immediately after  
14   only the smallest magnitude “ $n-k$ ” coefficients have been found. Moreover, the sorting  
15   procedure itself can be different depending on a comparison of “ $k$ ” to “ $n$ ” in order to  
16   minimize computations.

17          With reference to FIG. 16, there is shown a flowchart of a procedure that selects  
18   one of a number of techniques for finding a certain number “ $k$ ” of largest values out of a  
19   set of “ $n$ ” values. In a first step 271, execution branches to step 272 if “ $k$ ” is less than  $\frac{1}{2}$   
20   “ $n$ .” In step 272, execution branches to step 273 if “ $k$ ” is much less than  $\frac{1}{2}$  “ $n$ .” In step  
21   273, the first “ $k$ ” values are sorted to produce a list of “ $k$ ” sorted values, and then the last  
22   “ $n-k$ ” values are scanned for any value greater than the minimum of the sorted “ $k$ ”  
23   values. If a value greater than the minimum of the sorted “ $k$ ” values is found, then that

1 minimum value is removed and the value greater than the minimum value is inserted into  
2 the list of “k” sorted values. At the end of this procedure, the list of sorted “k” values  
3 will contain the maximum “k” values out of the original “n” values. A specific example  
4 of this procedure is described below with reference to FIG. 17.

5 In step 272, if “k” is not much less than  $\frac{1}{2}$  “n”, then execution branches to step  
6 274. In step 274, a bubble-sort procedure is used, including “k” bottom-up bubble-sort  
7 passes over the “n” values to put “k” maximum values on top of a sorting table. An  
8 example of such a bubble-sort procedure is listed below:

9

```
10 /* TABLE(0) to TABLE(n-1) INCLUDES n VALUES */  
11 /* MOVE THE k LARGEST OF THE n VALUES IN TABLE TO THE RANGE  
12 TABLE(0) TO TABLE(k-1) IN THE TABLE */  
13 /* k <=  $\frac{1}{2}$  n */  
14 FOR i=1 to k  
15 FOR j=1 to n-i  
16 IF (TABLE(n-j) > TABLE(n-j-1)) THEN(  
17     /* SWAP TABLE(n-j) WITH TABLE(n-j-1) */  
18     TEMP  $\leftarrow$  TABLE(n-j)  
19     TABLE(n-j)  $\leftarrow$  TABLE(n-j-1)  
20     TABLE(n-j-1)  $\leftarrow$  TEMP)  
21     NEXT j  
22     NEXT i  
23
```

1           In step 271, if “k” is not less than  $\frac{1}{2}$  “n”, then execution branches to step 275. In  
2       step 275, if “k” is much greater than  $\frac{1}{2}$  “n”, then execution branches to step 276. In step  
3       276, a procedure similar to step 273 is used, except the “n-k” minimum values are  
4       maintained in a sorted list, instead of the “k” maximum values. In step 276, the last “n-k”  
5       values are placed in the sort list and sorted, and then the first “k” values are scanned for  
6       any value less than the maximum value in the sorted list. If a value less than the  
7       maximum value in the sorted list is found, then the maximum value in the sorted list is  
8       removed, and the value less than this maximum value is inserted into the sorted list. At  
9       the end of this procedure, the values in the sorted list are the “n-k” smallest values, and  
10      the “k” values excluded from the sorted list are the “k” largest values.

11           In step 275, if “k” is not much greater than  $\frac{1}{2}$  “n”, then execution branches to step  
12       277. In step 277, a bubble-sort procedure is used, including “n-k” top-down bubble-sort  
13       passes over the “n” values to put “n-k” minimum values at the bottom of a sorting table.  
14       Consequently, the k maximum values will appear in the top “k” entries of the table. An  
15      example of such a bubble-sort procedure is listed below:

16  
17     /\* TABLE(0) to TABLE(n-1) INCLUDES n VALUES \*/  
18     /\* MOVE THE n-k SMALLEST OF THE n VALUES IN THE TABLE \*/  
19     /\* TO THE RANGE TABLE(k) TO TABLE(n-1) IN THE TABLE \*/  
20     /\* n > k >=  $\frac{1}{2}$  n \*/  
21     FOR i=1 to n-k  
22       FOR j=0 to n-i-1  
23         IF (TABLE(j) < TABLE(j+1)) THEN(

```
1      /* SWAP TABLE(j) WITH TABLE(j+1)*/  
2  
3      TEMP ← TABLE(j)  
4  
5      TABLE(j) ← TABLE(j+1)  
6  
7      TABLE(j+1) ← TEMP)  
8  
9      NEXT j  
10  
11     NEXT i  
12  
13
```

8       Turning now to FIG. 17, there is shown a flowchart of a procedure for finding up  
9       to a specified number "k" of largest magnitude AC DCT coefficients from a set of "n"  
10      coefficients, corresponding to the procedure of FIG. 16 for the case of  $k << \frac{1}{2}n$ . In a first  
11      step 281, a counter "i" is set to zero. In step 282, the next AC DCT coefficient is  
12      obtained from the input stream of original-quality MPEG-2 coded data. If an EOB  
13      marker is reached, as tested in step 283, then execution returns. In step 284, the counter  
14      "i" is compared to the specified number "k", and if "i" is less than "k", execution  
15      continues to step 285. In step 285, a coefficient index and magnitude for the AC DCT  
16      coefficient is placed on a sort list. In step 286, the counter "i" is incremented, and  
17      execution loops back to step 282.

18       Once the sort list has been loaded with indices and magnitudes for "k" AC DCT  
19      coefficients and one additional coefficient has been obtained from the input stream,  
20      execution branches from step 284 to step 287. In step 287 the list is sorted by magnitude,  
21      so that the minimum magnitude appears at the end of the list. Then in step 288 the  
22      coefficient magnitude of the current coefficient last obtained from the input stream is  
23      compared to the magnitude at the end of the list. If the coefficient magnitude of the

1 current coefficient is not greater than the magnitude appearing at the end of the list, then  
2 execution continues to step 289 to get the next AC DCT coefficient from the input  
3 stream. If an EOB marker is reached, as tested in step 290, then execution returns.  
4 Otherwise, execution loops back to step 288.

5 In step 288, if the magnitude of the current coefficient is greater than the  
6 magnitude at the end of the list, then execution branches to step 291. In step 291, the  
7 entry at the end of the list is removed. In step 292, a binary search is performed to  
8 determine the rank position of the magnitude of the current coefficient, and in step 293,  
9 the current coefficient index and magnitude are inserted into the list at the rank position.  
10 The list, for example, is a linked list in the conventional fashion to facilitate the insertion  
11 of an entry for the current coefficient at any position in the list. After step 293, execution  
12 loops back to step 288.

13 An approximation technique of coefficient magnitude classification can be used to  
14 reduce the computational burden of sorting by coefficient magnitude. A specific example  
15 is the use of hashing of the coefficient magnitude and maintaining lists of the indices of  
16 coefficients having the same magnitude classifications. As shown in FIG. 18, a hash  
17 table 300 is linked to hash lists 301 storing the indices of classified coefficients. As  
18 shown, the hash table 300 is a list of  $2^M$  entries, where “M” is three, and an entry has a  
19 value of zero if its associated list is empty, and otherwise the entry has a pointer to the  
20 end of the coefficients in its associated list. The lists shown in FIG. 18 have fixed  
21 memory allocations in which the pointers in the hash table also indicate the number of  
22 coefficient indices in the respective hash lists. Alternatively, the hash lists could be  
23 dynamically allocated and linked in the conventional fashion.

1 FIG. 19 shows a flowchart of a procedure for using the hash table 300 and hash  
2 lists 301 of FIG. 18 to perform a sort of “k” coefficients having approximately the largest  
3 magnitudes from a set of “n” coefficients. This approximation technique ensures that  
4 none of the “k” coefficients selected will have a magnitude that differs by more than a  
5 certain error limit from the smallest magnitude value of “k” coefficients having the  
6 largest magnitude. The error limit is established by the number of hash table entries, and  
7 it is the range of the magnitudes that can be hashed to the same hash table entry.

8 In a first step 311 in FIG. 19, the hash table is cleared. Then in step 312, the next  
9 AC DCT coefficient is obtained from the input stream. If an EOB marker is not reached,  
10 as tested in step 313, then execution continues to step 314. In step 314, a hash table  
11 index is stripped from the most significant bits (MSBs) of the coefficient magnitude. For  
12 the hash table in FIG. 18 having eight entries, the three most significant bits of the  
13 coefficient magnitude are stripped from the coefficient magnitude. This is done by a bit  
14 masking operation together with a logical arithmetic shift operation. Then in step 315,  
15 the coefficient index is inserted on the hash list of the indexed hash table entry. For  
16 example, the hash table entry is indexed to find the pointer to where the coefficient index  
17 should be inserted, and then the pointer in the hash table entry is incremented. After step  
18 315, execution loops back to step 312. Once all of the AC coefficients for the block have  
19 been classified by inserting them in the appropriate hash lists, an EOB marker will be  
20 reached, and execution will branch from step 313 to step 316.

21 Beginning in step 316, the hash table and hash lists are scanned to find  
22 approximately the “k” largest magnitude coefficients. The hash lists linked to the bottom  
23 entries of the hash table will have the indices for the largest magnitude coefficients. Each

1 hash list is scanned from its first entry to its last entry, so that each hash list is accessed as  
2 a first-in-first-out queue. Therefore, in each magnitude classification, the coefficient  
3 ordering in the output stream will be the same as the coefficient ordering in the input  
4 stream, and the approximation will have a “low pass” effect in which possibly some  
5 lower-frequency coefficients having slightly smaller magnitudes will be retained at the  
6 expense of discarding some higher-frequency coefficients having slightly larger  
7 magnitudes. (The approximation results from the fact that the last hash list to be scanned  
8 is not itself sorted, and to eliminate the error of the approximation, the last hash list to be  
9 scanned could be sorted.)

10 In step 316, a scan index “i” is set to  $2^M - 1$  in order to index the hash table  
11 beginning at the bottom of the table, and a counter “j” is set equal to “k” in order to stop  
12 the scanning process after finding “k” coefficients. Next, in step 317, the hash table is  
13 indexed with “i”. In step 318, if the indexed entry of the hash table is zero, then  
14 execution branches to step 319. In step 319, the procedure is finished if “i” is equal to  
15 zero; otherwise, execution continues to step 320. In step 320, the index “i” is  
16 decremented, and execution loops back to step 317.

17 If in step 318 the indexed hash table entry is not zero, then execution continues to  
18 step 321. In step 321, the next entry is obtained from the indexed hash list, and the  
19 coefficient index in the entry is used to put the indexed coefficient in the output stream.  
20 Then in step 322 the counter “j” is decremented, and in step 323 the counter “j” is  
21 compared to zero. In step 323, if the counter “j” is less than or equal to zero, then the  
22 procedure is finished. Otherwise, if the counter “j” is not less than or equal to zero in  
23 step 323, execution branches to step 324. In step 324, if the end of the hash list has not

- 1    been reached, execution loops back to step 321 to get the next entry in the hash list.  
2    Otherwise, if the end of the hash list has been reached, execution branches to step 319.

3              The FDSNR\_LM procedure, as described above, in general provides a significant  
4    improvement in peak signal-to-noise ratio (PSNR) over the FDSNR\_LP procedure when  
5    each procedure retains the same number of non-zero AC DCT coefficients. It has been  
6    found, however, that substantially more bits are required for the (run, level) coding of the  
7    non-zero AC DCT coefficients resulting from the FDSNR\_LM procedure than those  
8    resulting from the FDSNR\_LP procedure, provided that the same coefficient quantization  
9    and scanning method is used. Therefore, the FDSNR\_LM procedure provides at best a  
10   marginal improvement in rate-distortion (PSNR as a function of bit rate) over the  
11   FDSNR\_LP procedure unless the non-zero AC DCT coefficients for the FDSNR\_LM  
12   procedure are quantized, scanned, and/or (run, level) coded in a fashion different from the  
13   quantization, scanning, and/or (run, level) coding of the coefficients in the original  
14   MPEG-2 clip. A study of this problem resulted in a discovery that it is sometimes  
15   possible to reduce the number of bits for (run, level) coding of coefficients for an 8x8  
16   block including a given number of the non-zero largest magnitude AC DCT coefficients  
17   if additional coefficients are also (run, level) coded for the block.

18              The (run, level) coding of the non-zero AC DCT coefficients from the  
19   FDSNR\_LM procedure has been found to require more bits than from the FDSNR\_LP  
20   procedure due to an increased occurrence frequency of escape sequences for the (run,  
21   level) coding. The increased frequency of escape sequences is an indication that the  
22   statistical likelihood of possible (run, level) combinations for the non-zero AC DCT  
23   coefficients selected by the FDSNR\_LM procedure is different from the statistical

1 likelihood of possible (run, level) combinations for the non-zero AC DCT coefficients  
2 produced by the standard MPEG-2 coding process and in particular those selected by the  
3 FDSNR\_LP procedure.

4       The MPEG-2 coding scheme assigns special symbols to the (run, level)  
5 combinations that occur very frequently in ordinary MPEG-2 coded video. The most  
6 frequent (run, level) combinations occur for short run lengths (within the range of about  
7 0 to 5, where the run length can range from 0 to 63) and relatively low levels (about 1 to  
8 10, where the level can range from 1 to 2048). The most frequent of these special  
9 symbols are assigned variable-length code words (VLCs). If a (run, level) combination  
10 does not have such a VLC, then it is coded with an escape sequence composed of a 6-bit  
11 escape sequence header code word followed by a 6-bit run length followed by a 12 bit  
12 signed level. An escape sequence requires a much greater number of bits than the VLCs  
13 which have varying lengths depending on their relative frequency. In particular, the  
14 escape sequences each has 24 bits, and the variable-length code words have a maximum  
15 of 17 bits.

16       There are two (run, level) VLC tables in MPEG-2. The first coding table is  
17 designated TABLE 0, and the second is designated as TABLE 1. These tables specify  
18 the (run, level) combinations having VLCs. For each table, the (run, level) combinations  
19 represented by VLCs and the range of the VLC lengths are summarized below:

20

21       SUMMARY OF PROPERTIES OF DCT COEFFICIENT TABLE ZERO  
22       (Table Zero is Table B.14, p. 135 of ISO/IEC 13818-2 1996E)

23

<u>Run</u>	<u>Range of Levels</u>	<u>Range of Code Lengths</u>
2	0	1 to 40
3	1	1 to 18
4	2	1 to 5
5	3	1 to 4
6	4	1 to 3
7	5	1 to 3
8	6	1 to 3
9	7	1 to 2
10	8	1 to 2
11	9	1 to 2
12	10	1 to 2
13	11	1 to 2
14	12	1 to 2
15	13	1 to 2
16	14	1 to 2
17	15	1 to 2
18	16	1 to 2
19	17	1
20	18	1
21	19	1
22	20	1
23	21	1

1    22            1            14  
2    23            1            14  
3    24            1            14  
4    25            1            14  
5    26            1            14  
6    27            1            17  
7    28            1            17  
8    29            1            17  
9    30            1            17  
10   31            1            17

11

12    SUMMARY OF PROPERTIES OF DCT COEFFICIENT TABLE ONE

13    (Table One is Table B.15, p. 139 of ISO/IEC 13818-2 1996E)

14

15	<u>Run</u>	<u>Range of Levels</u>	<u>Range of Code Lengths</u>
16	0	1 to 40	3 to 16
17	1	1 to 18	4 to 17
18	2	1 to 5	6 to 14
19	3	1 to 4	6 to 14
20	4	1 to 3	7 to 13
21	5	1 to 3	7 to 14
22	6	1 to 3	8 to 17
23	7	1 to 2	8 to 13

1	8	1 to 2	8 to 13
2	9	1 to 2	8 to 14
3	10	1 to 2	8 to 14
4	11	1 to 2	9 to 17
5	12	1 to 2	9 to 17
6	13	1 to 2	9 to 17
7	14	1 to 2	10 to 17
8	15	1 to 2	10 to 17
9	16	1 to 2	11 to 17
10	17	1	13
11	18	1	13
12	19	1	13
13	20	1	13
14	21	1	13
15	22	1	14
16	23	1	14
17	24	1	14
18	25	1	14
19	26	1	14
20	27	1	17
21	28	1	17
22	29	1	17
23	30	1	17

2

3       The FDSNR\_LP procedure selected AC DCT coefficients have (run, level)  
4 symbol statistics that are similar to the statistics of ordinary MPEG-2 coded video, and  
5 therefore the FDSNR\_LP AC DCT coefficients have a similar frequency of occurrence  
6 for escape sequences in comparison to the ordinary MPEG-2 coded video. In contrast,  
7 the FDSNR\_LM procedure selects AC DCT coefficients resulting in (run, level)  
8 combinations that are less likely to be encountered in ordinary MPEG-2 coded video.  
9 This is due to two reasons. First, the FDSNR\_LM procedure selects AC DCT  
10 coefficients having the largest levels. Second, the FDSNR\_LM procedure introduces  
11 longer run lengths due to the elimination of coefficients over the entire range of  
12 coefficient indices. The result is a significantly increased rate of occurrence for escape  
13 sequences. Escape sequences form the most inefficient mode of coefficient information  
14 encoding in MPEG-2 incorporated into the standard so as to cover important but very  
15 rarely occurring coefficient information.

16       In order to improve the rate-distortion performance of the scaled-quality MPEG-2  
17 coded video resulting from the FDSNR\_LM procedure, the non-zero AC DCT  
18 coefficients selected by the FDSNR\_LM procedure should be quantized, scanned, and/or  
19 (run, level) coded in such a way that tends to reduce the frequency of the escape  
20 sequences. For example, if the original-quality MPEG-2 coded video was (run, level)  
21 coded using TABLE 0, then the largest magnitude coefficients should be re-coded using  
22 TABLE 1 because TABLE 1 provides shorter length VLCs for some (run, level)  
23 combinations having higher run lengths and higher levels. It is also possible that re-

1 coding using the alternate scan method instead of the zig-zag scan method may result in a  
2 lower frequency of occurrence for escape sequences. For example, each picture could be  
3 (run, level) coded for both zig-zag scanning and alternate scanning, and the scanning  
4 method providing the fewest escape sequences, or the least number of bits total, could be  
5 selected for the coding of the reduced-quality coded MPEG video.

6 There are two methods having general applicability for reducing the frequency of  
7 escape sequences resulting from the FDSNR\_LM procedure. The first method is to  
8 introduce a non-zero, “non-qualifying” AC DCT coefficient of the 8x8 block into the list  
9 of non-zero qualifying AC DCT coefficients to be coded for the block. In this context, a  
10 “qualifying” coefficient is one of the k largest magnitude coefficients selected by the  
11 FDSNR\_LM procedure. The non-qualifying coefficient referred to above, must be lying  
12 in between two qualifying AC DCT coefficients (in the coefficient scanning order) that  
13 generate the (run, level) combination causing the escape sequence. Moreover, this non-  
14 qualifying coefficient must cause the escape sequence to be replaced with two shorter  
15 length VLCs when the AC DCT coefficients are (run, level) coded. This first method has  
16 the effect of not only decreasing the number of bits in the coded reduced-quality MPEG  
17 video in most cases, but also increasing the PSNR.

18 The qualifying AC DCT coefficient causing the escape sequence that is first in the  
19 coefficient scanning order will be simply referred to as the first qualifying coefficient.  
20 The qualifying AC DCT coefficient causing the escape sequence that is second in the  
21 coefficient scanning order will be simply referred to as the second qualifying coefficient.  
22 For example, suppose the qualifying coefficients in zig-zag scan order for an 8x8 block  
23 include C<sub>51</sub> followed by C<sub>15</sub> having a level of 40. If only the qualifying coefficients were

1       (run, level) coded for the microblock,  $C_{15}$  would result in a run length of 3, because there  
2       are a total of three non-qualifying coefficients ( $C_{42}$ ,  $C_{33}$ , and  $C_{24}$ ) between  $C_{51}$  and  $C_{15}$  in  
3       the scan order. Therefore,  $C_{15}$  would have to be coded as an escape sequence, because a  
4       run of 3 and level of 40 does not have a special symbol. In this example, the escape  
5       sequence is in effect caused by a first qualifying coefficient, which is  $C_{51}$ , and a second  
6       qualifying coefficient, which is  $C_{15}$ . This escape sequence can possibly be eliminated  
7       say, if  $C_{24}$  is a non-zero, non-qualifying coefficient of the block,  $C_{24}$  has a level of 5 or  
8       less, and  $C_{24}$  is (run, level) coded together with the qualifying coefficients. For example,  
9       assuming that  $C_{24}$  has a level of 5, and using the MPEG-2 (run, level) coding TABLE 1,  
10      then  $C_{24}$  has a run length of two and is coded as the special symbol 0000 0000 1010 0s,  
11      where "s" is a sign bit, and  $C_{15}$  now has a run length of 0 and is coded as the special  
12      symbol 0000 0000 0010 00s. Such a consideration clearly applies to the rest of the non-  
13      zero non-qualifying coefficients lying in between the two qualifying coefficients  
14      producing the escape sequence. In the above example, these non-qualifying coefficients  
15      are  $C_{42}$  and  $C_{33}$ .

16           Whether or not an escape sequence can be eliminated from the (run, level) coding  
17      of the qualifying coefficients can be determined by testing a sequence of conditions. The  
18      first condition is that the second qualifying coefficient must have a level that is not  
19      greater than the maximum level of 40 for the special (run, level) symbols. If this  
20      condition is satisfied, then there must be a non-zero non-qualifying AC DCT coefficient  
21      that is between the first and second qualifying coefficients in the coefficient scanning  
22      order. If there is such a non-qualifying coefficient, then the combination of its level and  
23      the run length between the first qualifying coefficient and itself in the coefficient

1 scanning order must be one of the special (run, level) symbols. If so, then the  
2 combination of the level of the second qualifying coefficient and the run length between  
3 the non-qualifying coefficient and the second qualifying coefficient must also be a special  
4 (run, level) symbol, and if so, all required conditions have been satisfied. If not, then the  
5 conditions with respect to the non-qualifying coefficient are successively applied to any  
6 other non-zero non-qualifying AC DCT coefficient of the block lying in between the two  
7 qualifying coefficients, until either all conditions are found to be satisfied or all such non-  
8 qualifying coefficients are tested and failed. If there are sufficient computational  
9 resources, this search procedure should be continued to find all such non-qualifying  
10 coefficients that would eliminate the escape sequence, and to select the non-qualifying  
11 coefficient that converts the escape sequence to the pair of special symbols having  
12 respective code words that in combination have the shortest length.

13 A flow chart for a modified FDSNR\_LM procedure using the first method is  
14 shown in FIGS. 20 and 21. In a first step 331 of FIG. 20, the procedure finds up to "k"  
15 largest magnitude non-zero AC DCT coefficients (i.e., the "qualifying coefficients") for  
16 the block. (This first step 331 is similar to steps 261 to 265 of FIG. 15, as described  
17 above.) In step 332, (run, level) coding of the qualifying coefficients is begun in the scan  
18 order using the second coding table (Table 1). This (run, level) coding continues until an  
19 escape sequence is reached in step 333, or the end of the block is reached in step 336. If  
20 an escape sequence is reached, execution branches from step 333 to step 334. If the level  
21 of the second qualifying coefficient causing the escape sequence is greater than 40,  
22 execution continues from step 334 to step 336. Otherwise, execution branches from step  
23 334 to step 335 to invoke a subroutine (as further described below with reference to FIG.

1       21) to possibly include a non-zero non-qualifying AC DCT coefficient in the (run, level)  
2       coding to eliminate the escape sequence. The subroutine either returns without success,  
3       or returns such a non-qualifying coefficient so that the escape sequence is replaced with  
4       the two new (run, level) codings of the first qualifying coefficient and the non-qualifying  
5       coefficient and then the non-qualifying coefficient and the second qualifying coefficient.  
6       From step 335, execution continues to step 336. Execution returns from step 336 if the  
7       end of the block is reached. Otherwise, execution continues from step 336 to step 337, to  
8       continue (run, level) coding of the qualifying coefficients in the scan order using the  
9       second coding table (TABLE 1). This (run, level) coding continues until an escape  
10      sequence results, as tested in step 333, or until the end of the block is reached, as tested in  
11      step 336.

12           With reference to FIG. 21, there is shown a flow chart of the subroutine (that was  
13       called in step 335 of FIG. 20) for attempting to find a non-zero, non-qualifying AC DCT  
14       coefficient that can be (run, level) coded to eliminate an escape sequence for a qualifying  
15       coefficient. In a first step 341, the procedure identifies the first qualifying coefficient and  
16       the second qualifying coefficient causing the escape sequence. For example, the  
17       subroutine of FIG. 21 can be programmed as a function having, as parameters, a pointer  
18       to a list of the non-zero AC DCT coefficients in the scan order, an index to the first  
19       qualifying coefficient in the list, and an index to the second qualifying coefficient in the  
20       list. In step 342, the subroutine looks for a non-zero non-qualifying AC DCT coefficient  
21       between the first and the second qualifying coefficients in the scan order. For example,  
22       the value of the index to the first qualifying coefficient is incremented and compared to  
23       the value of the index for the second qualifying coefficient, and if they are the same, there

1 is no such non-qualifying coefficient. Otherwise, if the new coefficient pointed to (by  
2 incrementing the index of the first qualifying coefficient) is a non-zero coefficient then it  
3 becomes a candidate non-qualifying coefficient deserving further testing. If however the  
4 new coefficient pointed to (by incrementing the index of the first qualifying coefficient)  
5 has a value zero then it is not a candidate non-qualifying coefficient. If no such  
6 (candidate) non-qualifying coefficients are found, as tested in step 343, then execution  
7 returns from the subroutine with a return code indicating that the search has been  
8 unsuccessful. Otherwise, execution continues to step 344.

9 In step 344, the non-qualifying coefficient is (run, level) coded, to determine in  
10 step 345 whether it codes to an escape sequence. If it codes to an escape sequence, then  
11 execution loops back from step 345 to step 342 to look for another non-zero non-  
12 qualifying AC DCT coefficient in the scan order between the first and second qualifying  
13 coefficients. If it does not code to an escape sequence, then execution continues from  
14 step 345 to step 346. In step 346, the second qualifying coefficient is (run, level) coded,  
15 using the new run length, which is the number of coefficients in the scan order between  
16 the non-qualifying coefficient and the second qualifying coefficient. If it codes to an  
17 escape sequence, as tested in step 347, then execution loops back from step 347 to step  
18 342 to look for another non-zero non-qualifying AC DCT coefficient in the scan order  
19 between the first and second qualifying coefficients. If it does not code to an escape  
20 sequence, then execution continues from step 347 to step 348.

21 In step 348, execution returns with a successful search result unless a continue  
22 search option has been selected. If the continue search option has been selected, then  
23 execution branches from step 348 to step 349 to search for additional non-zero non-

1 qualifying AC DCT coefficients that would eliminate the escape sequence. In other  
2 words, steps 342 to 347 are repeated in an attempt to find additional non-zero non-  
3 qualifying AC DCT coefficients that would eliminate the escape sequence. If no more  
4 such non-qualifying coefficients are found, as tested in step 350, execution returns with a  
5 successful search result. Otherwise, execution branches from step 350 to step 351 to  
6 select the non-qualifying coefficient giving the shortest overall code word length and/or  
7 the largest magnitude for the best PSNR, and execution returns with a successful search  
8 result. For example, for each non-qualifying coefficient that would eliminate the escape  
9 sequence, the total bit count is computed for the (run, level) coding of the non-qualifying  
10 coefficient and the second qualifying coefficient. Then a search is made for the non-  
11 qualifying coefficient producing the smallest total bit count, and if two non-qualifying  
12 coefficients which produce the same total bit count are found, then the one having the  
13 largest level is selected for the elimination of the escape sequence.

14 A second method of reducing the frequency of occurrence of the escape  
15 sequences in the (run, level) coding of largest magnitude AC DCT coefficients for an 8x8  
16 block is to change the mapping of coefficient magnitudes to the levels so as to reduce the  
17 levels. Reduction of the levels increases the likelihood that the (run, level) combinations  
18 will have special symbols and therefore will not generate escape sequences. This second  
19 method has the potential of achieving a greater reduction in bit rate than the first method,  
20 because each escape sequence can now be replaced by the codeword for one special  
21 symbol, rather than by the two codewords as is the case for the first method. The second  
22 method, however, may reduce the PSNR due to increased quantization noise resulting  
23 from the process producing the lower levels. Therefore, if a desired reduction of escape

1 sequences can be achieved using the first method, then there is no need to perform the  
2 second method, which is likely to reduce the PSNR. If the first method is used but not all  
3 of the escape sequences have been eliminated, then the second method could be used to  
4 possibly eliminate the remaining escape sequences.

5 The mapping of coefficient magnitudes to the levels can be changed by decoding  
6 the levels to coefficient magnitudes, changing the quantization scale factor (qsi), and then  
7 re-coding the levels in accordance with the new quantization scale factor (qsi). The  
8 quantization scale factor is initialized in each slice header and can also be updated in the  
9 macroblock header on a macroblock basis. Therefore it is a constant for all blocks in the  
10 same macroblock. In particular, the quantization scale factor is a function of a  
11 q\_scale\_type parameter and a quantizer\_scale\_code parameter. If q\_scale\_type = 0, then  
12 the quantizer scale factor (qsi) is twice the value of q\_scale\_code. If q\_scale\_type =1,  
13 then the quantizer scale factor (qsi) is given by the following table, which is the right half  
14 of Table 7-6 on page 70 of ISO/IEC 13838-2:1996(E):

15

16	<u>quantizer_scale_code</u>	<u>quantization scale factor (qsi)</u>
17	1	1
18	2	2
19	3	3
20	4	4
21	5	5
22	6	6
23	7	7

1	8	8
2	9	10
3	10	12
4	11	14
5	12	16
6	13	18
7	14	20
8	15	22
9	16	24
10	17	28
11	18	32
12	19	36
13	20	40
14	21	44
15	22	48
16	23	52
17	24	56
18	25	64
19	26	72
20	27	80
21	28	88
22	29	96
23	30	104

2

3       In a preferred implementation, to reduce the coefficient levels, the quantization  
4       scale factor is increased by a factor of two, and the levels of the non-zero AC DCT  
5       coefficients are reduced by a factor of two, so long as the original value of the  
6       quantization scale factor is less than or equal to one-half of the maximum possible  
7       quantization scale factor. For `q_scale_type = 1`, a factor of two increase in the  
8       quantization scale factor (`qsi`) is most easily performed by a table lookup of a new  
9       quantization `_scale_code` using the following conversion table:

10

11	<u>Original quantization scale code</u>	<u>New quantization scale code</u>
12	1	2
13	2	4
14	3	6
15	4	8
16	5	9
17	6	10
18	7	11
19	8	12
20	9	14
21	10	16
22	11	17
23	12	18

1	13	19
2	14	20
3	15	21
4	16	22
5	17	24
6	18	25
7	19	26
8	20	27
9	21	28
10	22	29
11	23	30
12	24	31

14 V. Trick Mode Files

15 In a preferred method for generation of trick mode files, the quantization scale  
16 factor is adjusted in order to achieve a desired reduction in the escape sequence  
17 occurrence frequency resulting from the modified FDSNR\_LM procedure, and the  
18 number (k) of largest magnitude coefficients is adjusted in order to achieve a desired  
19 reduction in bit rate. A specific implementation is shown in the flow chart of FIGS. 22-  
20 23. In a first step 361, the number (k) of largest magnitude AC coefficients per 8x8 block  
21 is initially set to a value of 9, and the quantization scaling factor (QSF) is initially set to a  
22 value of 2. Then conversion of the I frames of an original-quality MPEG-2 coded video  
23 clip to a lower quality level begins. When a picture header is encountered in step 362,

1 indicating the beginning of a new I frame, execution continues to step 363. In step 363,  
2 execution branches depending on the value of the intra\_vlc\_format parameter in the  
3 picture header of the original-quality MPEG-2 coded video clip. This value is either 0,  
4 indicating that the first (run, level) coding table (TABLE 0) was used for coding the  
5 picture, or 1, indicating that the second (run, level) coding table (TABLE 1) was used for  
6 coding the picture. In either case, the down scaled quality picture will be coded with the  
7 second (run, level) coding table. If the intra\_vlc\_format parameter is equal to 0 execution  
8 continues from step 363 to step 364 where TABLE 0 is read in for (run, level) symbol  
9 decoding in the original-quality MPEG-2 coded clip. Otherwise, if the intra\_vlc\_format  
10 parameter is equal to 1, then execution continues from step 363 to step 365 where  
11 TABLE 1 is read in for (run, level) symbol decoding in the original-quality MPEG-2  
12 coded clip.

13 After steps 364 and 365, execution continues to step 366. In step 366, the  
14 modified FDSNR\_LM procedure is applied to the 8x8 blocks of the current slice, using  
15 the adjusted quantization scale index, if the adjusted quantization scale index is less than  
16 the maximum possible quantization scale index. In step 367, execution loops back to step  
17 362 to continue 8x8 block conversion until a new slice header is encountered, indicating  
18 the beginning of a new slice. Once a new slice is encountered, execution continues from  
19 step 367 to step 368. In step 368, the average escape sequence occurrence frequency per  
20 block for the last slice is compared to a threshold TH1. If the escape sequence  
21 occurrence frequency is greater than the threshold, then execution branches to step 369.  
22 In step 369, if the quantization scaling factor (QSF) is less than or equal to a limit value

1 such as 2, then execution branches to step 370 to increase the quantization scaling factor  
2 (QSF) by a factor of two.

3 In step 368, if the escape sequence occurrence frequency is not greater than the  
4 threshold TH1, then execution continues to step 371 of FIG. 23. In step 371, the average  
5 escape sequence occurrence frequency per 8x8 block for the last slice is compared to a  
6 threshold TH2. If the escape sequence occurrence frequency is less than the threshold  
7 TH2, then execution branches to step 372. In step 372, if the quantization scaling factor  
8 (QSF) is greater than or equal to a limit value such as 2, then execution branches to step  
9 373 to decrease the quantization scaling factor (QSF) by a factor of two. After step 373,  
10 and also after step 370 of FIG. 22, execution continues to step 374 of FIG. 23. In step  
11 374, execution continues to step 375 if a backtrack option has been selected. In step 375,  
12 re-coding for the last slice is attempted using the adjusted quantization scale factor. The  
13 new coding, or the coding that gives the best results in terms of the desired reduction of  
14 escape sequence occurrence frequency, is selected for use in the scaled quality picture.  
15 After step 375, execution continues to step 376. Execution also continues to step 376  
16 from: step 369 in FIG 22 if the quantization scaling factor (QSF) is not less than or equal  
17 to 2; step 371 in FIG 23 if the escape sequence occurrence frequency is not less than the  
18 threshold TH2; step 372 in FIG 23 if the quantization scaling factor (QSF) is not greater  
19 than or equal to 2; and from step 374 in FIG 23 if the backtrack option has not been  
20 selected.

21 In step 376, the average bit rate of the (run, level) coding per 8x8 block for at  
22 least the last slice is compared to a high threshold TH3. Preferably this average bit rate is  
23 a running average over the already processed segment of the current scaled quality I-

1 frame, and the high threshold TH3 is selected to prevent video buffer overflow in  
2 accordance with the MPEG-2 Video Buffer Verifier restrictions. If the average bit rate  
3 exceeds the high threshold TH3, then execution continues to step 377, where the number  
4 (k) of non-zero largest magnitude AC coefficients per 8x8 block is compared to a lower  
5 limit value such as 6. If the number (k) is greater than or equal to 6, then execution  
6 continues to step 378 to decrement the number (k).

7 In step 376, if the average bit rate is not greater than the threshold TH3, then  
8 execution continues to step 379. In step 379, the average bit rate is compared to a lower  
9 threshold TH4. If the average bit rate is less than the threshold TH4, then execution  
10 branches from step 379 to step 380, where the number (k) of non-zero largest magnitude  
11 AC DCT coefficients per 8x8 block is compared to a limit value of 13. If the number (k)  
12 is less than or equal to 13, then execution continues to step 381 to increment the number  
13 (k). After step 378 or 381, execution continues to step 382. In step 382, execution  
14 continues to step 383 if a backtrack option is selected. In step 383, an attempt is made to  
15 re-code the last slice for the scaled quality picture using the adjusted value of the number  
16 (k) of non-zero largest magnitude AC DCT coefficients per block. After step 383,  
17 execution loops back to step 362 of FIG. 22 to continue generation of the scaled quality  
18 clip. Execution also loops back to step 362 of FIG. 22 after: step 377 if the value of (k) is  
19 not greater than or equal to 6; step 379 if the average bit rate is not less than the threshold  
20 TH4; step 380 if the value of (k) is not less than or equal to 13; and step 382 if the  
21 backtrack option has not been selected. Coding of the scaled quality clip continues until  
22 the end of the original quality clip is reached in step 384 of FIG. 22, in which case  
23 execution returns.

1           In a preferred implementation, a fast forward trick mode file and a fast reverse  
2 trick mode file are produced from an original-quality MPEG-2 coded video main file  
3 when the main file is ingested into the video file server. As shown in FIG. 24, a volume  
4 generally designated 390 is allocated to store the main file 391. The volume 390 includes  
5 an allocated amount of storage that exceeds the real file size of the main file 391 in order  
6 to provide additional storage for meta-data 392, the fast forward trick file 393, and the  
7 fast reverse trick file 394. The trick files are not directly accessible to clients as files;  
8 instead, the clients may access them through trick-mode video service functions. With  
9 this strategy, the impact on the asset management is a minimum. No modification is  
10 needed for delete or rename functions.

11           Because the volume allocation is done once for the main file and its fast forward  
12 and fast reverse trick mode files, there is no risk of lack of disk space for production of  
13 the trick files. The amount of disk blocks to allocate for these files is computed by the  
14 video service using a video service parameter (vsparams) specifying the percentage of  
15 size to allocate for trick files. A new encoding type is created in addition to types RAW  
16 for direct access and MPEG2 for access to the main file. The new encoding type is called  
17 EMPEG2, for extended MPEG2, for reference to the main file plus the trick files. The  
18 video service allocates the extra file size only for these files.

19           For the transfer of these files to archive or to another video file server, it would be  
20 useful to transfer all the data even if it is a non-standard format. For the FTP copy-in, a  
21 new option is added to specify if the source is in the EMPEG2 format or if it is a standard  
22 MPEG2 file. In the first case, the copy-in should provide the complete file 390. In the  
23 second case, the video service allocates the extra size and the processing is the same as

1 for a record. For the copy-out, the same option can be used to export the complete file  
2 390 or only the main part 391. The archiving is always done on the complete file 390.

3       The trick mode file production is done by a new video service procedure. This  
4 procedure takes as input the speed-up factor (or the target trick mode file size) along with  
5 the number of freeze (P or B) frames to insert in between the scaled I frames and then  
6 generates both the fast forward file 393 and the fast reverse file 394 for this speed-up  
7 factor (or target trick mode file size) and with the specified number of interleaving freeze  
8 frames. Since the bandwidth of the original clip (in the main file) and the bandwidths of  
9 the two trick mode clips (in the fast forward and fast reverse files) are the same, the  
10 speed-up factor and the target trick mode file size are equivalent pieces of information. A  
11 default speed-up factor (system parameter) can be used. The main file is read and the  
12 trick mode files are produced. If a trick mode file already exists with the same speed-up  
13 factor, it is rewritten or nothing is done depending on an option. Multiple trick mode  
14 files could be created with different speed-up factors. But it is preferred to permit only  
15 one set of fast forward and fast reverse trick mode files to be produced at a time (i.e., no  
16 parallel generation with different speed-up factors). The current speed-up factor is a  
17 parameter within the video service parameters (vsparams).

18       As stated above another parameter to be provided to the video service procedure  
19 in charge of trick mode file generation is the number of freeze frames to be inserted in  
20 between consequent scaled I frames. The preferred values for this parameter are 0 and 1,  
21 although other positive integer values greater than 1 are also possible. The inclusion of  
22 freeze frames due to their very small sizes spare some bandwidth which can then be used  
23 to improve the quality of scaled I frames. Hence, the freeze frames in this context provide

1 a mechanism to achieve a trade-off between the scaled I frame quality and the temporal  
2 (motion) sampling. Depending on the speed-up factor (or the target trick mode file size)  
3 and also the number of interleaving freeze frames to be inserted, the video service  
4 procedure in charge of trick mode file generation determines a sub-sampling pattern  
5 (closest to uniform) to choose the original I frames which will be scaled and included in  
6 the trick mode files. For example, the case of an original clip with 10 frames per GOP, a  
7 trick mode file size which is 10% of the main file together with 0 freeze frames, implies  
8 the use of all original I frames for being scaled and included in the trick mode file. This  
9 will typically result in a low quality scaling. As another example, the case of an original  
10 clip with 10 frames per GOP, a trick mode file size which is 10% of the main file  
11 together with 1 freeze frame, implies the use of a 2 to 1 (2:1) sub-sampling on the  
12 original I frames which will choose every other original I frame for being scaled and  
13 included in the trick mode file.

14 FIG. 25 is a more detailed diagram of the volume 390, showing additional meta-  
15 data and related data structures. The Inode 401 includes 4 disk blocks containing a file-  
16 system oriented description of the file. The meta-data (MD) directory 402 includes 4  
17 disk blocks describing each entry of the meta-data area 392. The entries of the meta-data  
18 area 392 include a description of the MPEG-2 meta-data 403, a description of the trick  
19 files header meta-data 404, and a description of the GOP index meta-data 405. The  
20 MPEG-2 meta-data 403 includes 15 disk blocks maximum.

21 The trick files header 404 includes 1 disk block, which specifies the beginning of  
22 free area (end of last trick file) in blocks, the number of trick files couple (FF FR), and  
23 for each trick file, a speed-up factor, a block address of the GOP index, a block address of

1 the trick file forward, a byte length of the trick file forward, a block address of the trick  
2 file reverse, a byte length of the trick file reverse, a frames number of the trick file, and a  
3 number of GOP of each trick files.

4           The GOP index includes 2024 disk blocks. The GOP index specifies, for each  
5 GOP, a frame number, a pointer to the MPEG-2 data for the GOP in the main file, and  
6 various flags and other attributes of the GOP. The flags indicate whether the GOP entry  
7 is valid and whether the GOP is open or closed. The other attributes of the GOP include  
8 the maximum bit rate, the average bit rate, the AAU size in bytes, the APU duration in  
9 seconds, the audio PES packet starting locations, the AAU starting locations, the AAU  
10 PTS values, and the decode time stamp (DTS) and the value of the program clock  
11 reference (PCR) extrapolated to the first frame of the GOP. The size of all the data  
12 preceding the main file is, for example, 1 megabyte.

13           There is one GOP index 406 for both the fast forward file 393 and the fast reverse  
14 file 394. The GOP index 406 of the trick files is different than the GOP index 405 of the  
15 main file. The GOP index 406 of the trick files contains, for each GOP, the byte offset in  
16 the trick file forward of the TS packet containing the first byte of the SEQ header, the  
17 frame number in the fast forward file of the GOP (the same value for the fast reverse file  
18 can be computed from this value for the fast forward file), the frame number in the  
19 original file of the first frame of the GOP, and the byte offset in the original file of the  
20 same frame (to resume after fast forward or reverse without reading the main GOP  
21 index).

22           The GOP index 405 for the main file and the GOP index 406 for the fast forward  
23 and fast reverse trick files provide a means for rapidly switching between the normal

1 video-on-demand play operation during the reading of the main file, and the fast-forward  
2 play during the reading of the fast-forward file, and the fast-reverse play during the  
3 reading of the fast-reverse file. For example, FIG. 26A illustrates the read access to  
4 various GOPs in the main file, fast forward file, and fast reverse file, during a play  
5 sequence listed in FIG. 26B. Due to the presence of down-scaled I frames and possibly  
6 present consequent freeze frames in the trick mode files, the video buffer verifier (VBV)  
7 model for a trick mode file is different than the VBV model of the main file.  
8 Consequently, the mean video decoder main buffer fullness levels can be significantly  
9 different for these files. For example, a transition from the main file to one of the trick  
10 files will usually involve a discontinuity in the mean video decoder main buffer fullness  
11 level, because only the I frames of the main file correspond to frames in the trick files,  
12 and the corresponding I frames have different bit rates when the trick mode I frames are  
13 scaled down for a reduced bit rate. An instantaneous transition from a trick file back to  
14 the main file may also involve a discontinuity especially when freeze frames are inserted  
15 between the I frames for trick mode operation. To avoid these discontinuities, the  
16 seamless splicing procedure of FIGS. 3 to 6 as described above is used during the  
17 transitions from regular play mode into trick mode and similarly from trick mode back  
18 into the regular play mode. Through the use of the seamless splicing procedure to modify  
19 the video stream content, for example for the “Seamless Splice” locations identified in  
20 FIG. 26A, the video decoder main buffer level will be managed so as to avoid both  
21 overflows and underflows leading to visual artifacts.

22 It is desired to copy in and out of the volume 390 with or without the meta-data  
23 392 and the trick files 393, 394. This is useful to export and/or import complete files

1 without regenerating the trick files. The file encoding type is now recognized as a part of  
2 the volume name. Therefore there can be multiple kinds of access to these files. The  
3 read and write operations are done by derivations of the class file system input/output  
4 (FSIO) which takes into account the proper block offset of the data to read or write.  
5 There is one derivation of FSIO per encoding type, providing three different access  
6 modes. EMPEG2, MPEG2, and RAW. EMPEG2 accesses the whole volume from the  
7 beginning of the meta-data array, and in fact provides access to the entire volume except  
8 the inode 401, but no processing is done. MPEG2 access only the main part of the asset  
9 with MPEG processing, including file analysis and meta-data generation in a write  
10 access. RAW access only the main part of the asset without processing. These access  
11 modes are operative for read and write operations for various access functions as further  
12 shown in FIG. 27.

13 During a record operation, the video service allocates a volume and computes the  
14 number of blocks to allocate using the volume parameter giving the percentage to add for  
15 the trick files. Then, the size in blocks given to the stream server is the main part size  
16 only without the extension for the trick files. This avoids using the reserved part of the  
17 volume when the effective bit rate is higher than the requested bit rate. At the end of a  
18 record operation or an FTP copy-in operation, the video service calls a procedure  
19 CMSPROC\_GETATTR, and the stream server returns the actual number of bytes  
20 received and the actual number of blocks used by the main file plus the meta-data. The  
21 same values are returned for both MPEG2 and EMPEG2 files. The video service  
22 computes again the file extension to manage the trick files and adjust the number of  
23 allocated blocks.

1 Both trick files, forward and reverse, are generated by the same command. First,  
2 the trick file forward is generated by reading the main file. The trick file GOP index is  
3 concurrently built and kept in memory. During this generation, only the video packets  
4 are kept. PCR, PAT and PMT will be regenerated by the MUX in play as for any other  
5 streams. The audio packets are discarded. This ensures that there is enough stuffing  
6 packets for the PCR reinsertion. For this, a stuffing packet is inserted every 30  
7 milliseconds.

8 Then using the GOP index, the trick file forward is read GOP by GOP in reverse  
9 order to generate the trick file reverse. The same GOPs are present in both files. The  
10 only modification done is an update of the video PTS, which must be continuous. Then,  
11 the GOP index is written on disk. This avoids reading again the file while generating the  
12 second trick file. The GOP index size is: 24 times the GOP number. In the worst case  
13 (the file is assumed not to be 1 frame only), there are 2 frames per GOP and 30 frames  
14 per second. So for 1 hour in fast forward, the GOP index size is:  $(24 \times 3600 \times 30) / 2 =$   
15 1296000 bytes. This will be the case for a 4 hour film played at 4 times the normal  
16 speed. Therefore, this GOP index can be kept in memory during the trick file generations  
17 without risk of memory overflow.

18 The read and write rates are controlled to conserve bandwidth on the cached disk  
19 array. The bandwidth reserved for these operations is a parameter given by the video  
20 service. It is a global bandwidth for both read and writes. The number of disk I/Os per  
21 second is counted so as not to exceed this bandwidth.

22 The trick files' header update is done once when both the fast forward and fast  
23 reverse trick files and the GOP index have been successfully written.

1 Playing a file is done with the CM\_MpegPlayStream class. Fast forward  
2 (reverse) can only be requested when the stream is in the paused state. The current frame  
3 on which the stream is paused is known from the MpegPause class. This frame is located  
4 in the GOP index of the trick file. Then the clip start point and length are modified in the  
5 Clip instance with the trick file position computed from the beginning of the clip. So, the  
6 Clip class handles these trick files in a manner similar to the main file. The current  
7 logical block number is updated with the block address in the trick file recomputed from  
8 the beginning of the main clip. In fact, a seek is performed in the trick file as it was part  
9 of the main file, which is totally transparent for the ClipList and Clip classes. The  
10 transition from fast forward to pause is handled in a similar fashion. The clip start and  
11 length and the logical block number are again updated. The smooth transitions from  
12 pause to fast forward and from fast forward to pause are done in the same way as for  
13 regular play. There is a splicing from the pause stream to the play stream.

14 The class hierarchy for trick file handling is shown in FIG. 28. The MpegFast,  
15 MpegFastForward and MpegFastReverse class handles the GOP generation from the  
16 initial file. This is the common procedure for building the GOP whatever the source and  
17 the destination are. RealTimeFastFwd and RealTimeFastRev are the classes instantiated  
18 when a real time fast forward (reverse) has to be done. They manage the real-time buffer  
19 flow to the player. There is a derivation of the methods takeBuffer and returnBuffer  
20 which use the base class to build the GOP in the buffer to be played. The main file  
21 access is done using a buffer pool.

22 TrickFilesGenerate is the class instantiated to generate trick files forward and  
23 reverse. It inherits from TrickFilesAccess the methods for reading the original file into

1 some buffers and for writing the trick file and its meta-data. It inherits from  
2 MpegFastForward the methods for building the GOP and for managing the advance in  
3 the file.

4 The computation of the next 1 frame to play is done by MpegFast,  
5 MpegFastForward and RealTimeFastFwd. When a trick file generation command is  
6 invoked, a thread is created and started and the generation itself is done off-line. A call-  
7 back is sent to the video service when the generation is completed. The class  
8 TrickFilesGenerate generates the trick file forward, and then, using the GOP index built  
9 in memory, the class TrickFilesGenerate generates the trick file reverse.

10 When there is a transition from play to pause, the only latency issue is related to  
11 the buffer queue handled by the player and to the GOP size. The stream can build  
12 immediately the active pause GOP, and then this GOP will be sent at the end of the  
13 current GOP with a splicing between these two streams.

14 When there are transitions from pause to regular play or fast forward and fast  
15 reverse, a seek in the file is done. This means that the current buffer pool content is  
16 invalidated and the buffer pool is filled again. Play can start again while the buffer pool  
17 is not completely full, as soon as the first buffer is read. The buffer pool prefilling can  
18 continue as a background process. The issue here is that there is a risk to generate an  
19 extra load on the cached disk array as well as on the stream server side when the buffer  
20 pool is being prefilled.

21 To avoid too frequent transitions from play to fast forward and fast reverse, there  
22 is a limitation of the number of requests per second for each stream. This limitation is  
23 part of the management of the video access commands. A minimum delay between two

1 commands is defined as a parameter. If the delay between a request and the previous one  
2 is too small, the request is delayed. If a new request is received during this delay, the  
3 new request replaces the waiting one. So the last received request is always executed.

4 The video service parameters file (vsparams) contains these new parameters for  
5 the trick mode files:

6 TrickFileExtensionSize:<percent>:

7 DefaultFastAcceleration:<acceleration>:

8 DMtrickFileGen:<mask of reserved DM> (This parameter is a mask of the stream  
9 servers that can be chosen to perform the trick file generation. The default value is  
10 0xffffc: all of the stream servers.)

11 DMtrickFileGenBW:<bandwidth used for trick file generation> (This parameter  
12 is the value of the bandwidth effectively used by the stream server for the trick files  
13 generation.)

14 The video service routines are modified to operate upon the EMPEG2 files, and in  
15 particular to compute the size of the EMPEG2 files, to allocate the volume for the main  
16 file and the trick files, and to generate the trick files. The volume creation functions  
17 (VAPP) and volume access functions (RRP) use the EMPEG2 files in the same way as  
18 MPEG2 files. This means that an MPEG2 volume is created on the stream server. Both  
19 MPEG2 and EMPEG2 files can be used in the same session or play-list. The session  
20 encoding type is MPEG2. In record (or copy-in), the number of blocks allocated for an  
21 EMPEG2 file is computed using the percentage of size to add. At the end of record (or  
22 copy-in), the number of blocks is adjusted using the number of blocks returned by the  
23 stream server (by CMSPROC\_GETATTR) and adding the percentage for trick files. The

1 trick files validity and generation date are stored by the video service in the asset  
2 structure. The bandwidth allocated to the TrickFilesGenerate command is defined in the  
3 video service parameters (vsparams or vssiteparams). The selection of a stream server to  
4 generate the trick files takes into account this bandwidth only. If preferred stream servers  
5 are specified in vsparams (or vssiteparams), then the selected stream server will be one of  
6 these specified stream servers.

7 In a preferred implementation of the video service software, a new encoding type  
8 is created. The encoding type enum becomes:

```
9     enum encoding-t{  
10         ENC_UNKNOWN    = 0,          /* unknown format */  
11         ENC_RAW        = 1,          /* uninterpreted data */  
12         ENC_MPEG1      = 2,          /* constrained MPEG1 */  
13         EMC_MPEG       = 3,          /* generic MPEG */  
14         ENC_EMPEG2     = 4,          /* MPEG2 with trick files extension */  
15     };  
16
```

17 - The encoding information accessible by VCMP\_EXTENDEDINFO includes  
18 information about trick files:

```
19  
20     struct trickFileInfo_t{  
21         ulong_t        generationDate;      /* date/time of the generation of the trick  
22                                         files */  
23         rate_factor_t   acceleration;        /* acceleration factor */
```

```

1      ulong_t      framesNumber;      /* frames number in each trick file (FWD and
2                                REV) */
3      ulong_t      gopNumber;        /* GOP number of each file */
4      };
5
6  struct EMPEG2info_t{
7      MPEG2info_t      MPEG2info;
8      trickFilesInfo_t      trickFiles<>;
9      };
10
11 union encodingInfo_t switch (encoding-t enc){
12     case ENC_MPEG:
13         MPEG2info_t      MPEG2info;
14     case ENC_EMPEG2:
15         EMPEG2info_t      EMPEG2info;
16     default:
17         void;
18     };
19
20 The video service software includes a new procedure (VCMP_TRICKFILESGEN) for
21 trick file generation, which uses the following structures:
22
23 struct VCMPtrickgenres_t{

```

```
1      VCMPstatus_t      status;
2      tHandle_t         handle;
3      };
4
5  struct VCMPtrickfilesargs_t{
6      name_t      clipname;
7      bool_t       overwriteIfExists;
8      rate_factor_t acceleration;
9  };
10
11 VCMPtrickgenres_t      VCMP_TRICKFILESGEN (VCMPtrickfilesargs_t)  =
12 36,
```

13

14 If the trick files already exist and if the boolean overwriteIfExists is true, then the  
15 trick files are generated again, in the other case nothing is done. Acceleration is the  
16 acceleration as defined and used for the controlled speed play function. It is a percentage  
17 of the normal speed, it must be greater than 200 and smaller than 2000. The special value  
18 of 0 can be used to generate files with the default acceleration defined in vssiteparams.  
19 The procedure starts the generation process. The completion is notified by a callback.

20 The video service includes a new option to copy-in and copy-out. The option is  
21 added to allow a user to copy all the file or the main asset only. For compatibility with  
22 old client programs, the following new procedures are added:

23

1   VCMPcopyres\_t     VCMP\_FULL\_COPYIN    (copyinargs2\_t)     = 37,  
2   VCMPcopyres\_t     VCMP\_FULL\_COPYOUT    (copyoutargs2\_t)    = 38,  
3  
4   These new procedures have the same interface as the already existing one, but are used to  
5   copy-in the complete file: meta-data + asset + trick files.

6  
7       The video service includes a new procedure  
8       VCMP\_TRICKFILESGENCOMPLETED, which uses the following structures:

9  
10      struct VCMPtrickfilecomplete\_t{  
11           tHandle\_t                  handle;  
12           VCMPstatus\_t              status;  
13      };  
14  
15      VCMPstatus\_t TRICKFILESGENCOMPLETED (VCMPtrickfilecomplete\_t) = 10,

16  
17       The video service includes new procedures added for handling trick mode  
18       generation arguments and using the following structures:

19  
20      struct cms\_trick\_gen\_args {  
21           Handle\_t                 Vshandle;  
22           name\_t                 name;  
23           bool\_t                 overwriteIfExists;

```
1      rate_factor_t      acceleration;
2      bandwidth_t       reservedBw;
3  };
4
5  cms_status   CMSPROC_GEN_TRICK_FILES (cms_trick_gen_args)      = 34,
6
7  struct trick_gen_completed_args {
8      Handle_t          Vshandle;
9      cms_status        status;
10 };
11 void CTLPROC_TRICKGENCOMPLETED (trick_gen_completed_args)      = 8,
```

12  
13 The video service includes the following option to force the regeneration of trick  
14 files even if they exist:

15 nms\_content -gentrick <name> [<-f>] [acceleration]

16 Without this option, an error code is returned if the trick files exist. “Acceleration” is an  
17 acceleration factor. If it is not present, the default value is taken from vsparams.

18 The video services include an encoding information access function (nms\_content  
19 -m). This function produces displayed output containing, for each trick file generated,  
20 the acceleration, the generation date and time, the number of frames, and the number of  
21 GOPs.

22 For the use of an FTP copy function with the trick files, the following new  
23 commands are added:

1  
2 nms\_content –copyinfull <same arguments as –copyin>  
3 nms\_content –copyoutfull <same arguments as –copyout>  
4

5 VI. Reduction of MPEG-2 Transport Stream Bit Rate for Combining Multiple  
6 MPEG-2 Transport Streams

7 Another application of the SNR scaling achieved by the invention is to reduce the  
8 bit rate of an MPEG-2 transport stream in order to allow combining multiple MPEG-2  
9 transport streams to match a target bit rate for a multiple program transport stream. For  
10 example, FIG. 29 shows a system for combining an MPEG-2 audio-visual transport  
11 stream 411 with an MPEG-2 closed-captioning transport stream 412 to produce a  
12 multiplexed MPEG-2 transport stream 413. In this case, the closed captioning transport  
13 stream 412, containing alphanumeric characters and some control data instead of audio-  
14 visual information, has a very low bit rate compared to the audio-visual transport stream  
15 411. Assuming that the target bit rate for the multiplexed transport stream 413 is the  
16 same as the bit rate of the audio-visual transport stream 411, there need be only a slight  
17 decrease in the bit rate of the audio-visual transport stream, and this slight decrease can  
18 be obtained by occasionally removing one non-zero AC DCT coefficient per 8x8 block.  
19 Therefore, in the system of FIG. 29, the audio-visual transport stream 411 is processed by  
20 a program module 414 for selective elimination of non-zero AC DCT coefficients to  
21 slightly reduce the average bit rate of this transport stream. A transport stream  
22 multiplexer 415 then combines the modified audio-visual transport stream with the closed

1 captioning transport stream 412 to produce the multiplexed MPEG-2 transport stream  
2 413.

3 In order to determine whether or not any non-zero AC DCT coefficient should be  
4 eliminated from a next 8x8 block in the audio-visual transport stream 411, a module 421  
5 is executed periodically to compute a desired bit rate change in the audio-visual transport  
6 stream 411. For example, respective bit rate monitors 416, 417 may measure the actual  
7 bit rate of the audio-visual transport stream 411 and the closed captioning transport  
8 stream 412. Alternatively, if it is known precisely how these transport streams are  
9 generated, presumed values for the bit rates of these transport streams may be used in lieu  
10 of measured bit rates. The computation of the desired bit rate change also includes the  
11 desired bit rate 418 for the multiplexed MPEG-2 transport stream, and a bit rate 419 of  
12 multiplexer overhead, representing any net increase in bit rate related to the multiplexing  
13 of the audio-visual transport stream 411 with the closed captioning transport stream 412.  
14 An adder/subtractor 420 combines the various bit rate values from the inputs 416, 417,  
15 418, and 419 to compute the desired bit rate change in the audio-visual transport stream  
16 411. From the adder/subtractor 420 output, the module 421 converts the desired change  
17 in bit rate to a desired number of bits to be removed per computational cycle (e.g., per  
18 millisecond). This number of bits to be removed per computational cycle is received in  
19 an adder/subtractor 422, and the output of the adder/subtractor is received in an integrator  
20 423. A limiter 424 takes the sign (positive or negative) of the integrated value to produce  
21 a flag indicating whether or not one non-zero AC DCT coefficient should be removed  
22 from the coefficients for the next 8x8 block, assuming that the next block has at least one  
23 non -zero AC DCT coefficient. (Alternatively, a non-zero AC DCT coefficient could be

1 removed only if the 8x8 block has more than a predetermined fraction of the average  
2 number of AC DCT coefficients per 8x8 block.) The particular non-zero AC DCT  
3 coefficient to remove in each case can be selected using any of the methods discussed  
4 above with reference to FIGS. 14, 15, or FIG. 20. For example, the coefficient to remove  
5 could be the last non-zero AC DCT coefficient in the scan order. Alternatively, the non-  
6 zero AC DCT coefficient having the smallest magnitude could be removed so long as its  
7 removal does not cause an escape sequence.

8 When the module 414 removes a non-zero AC DCT coefficient from a 8x8 block,  
9 it sends the number of bits removed to the adder/subtractor 422. In a preferred  
10 implementation, the operations of the adder/subtractor 422, integrator 423, and limiter  
11 424 are performed by a subroutine having a variable representing the integrated value.  
12 During each computational cycle, the variable is incremented by the number of bits to be  
13 removed per computational interval, and whenever the module 414 removes a non-zero  
14 AC DCT coefficient from a 8x8 block of the audio-visual transport stream, the variable is  
15 decremented by the number of bits removed.

16 Although the system in FIG. 29 has been described for achieving a slight  
17 reduction in bit rate of the MPEG-2 audio-visual transport stream 411 for combining  
18 multiple transport streams to produce a multiple program MPEG-2 transport stream, it  
19 should be apparent that it could be used for obtaining relatively large reductions in bit  
20 rate. In this case, the module 414 would use the procedure of FIGS. 14, 15 or preferably  
21 FIG. 20, and a multi-level comparator 424 would be used instead of a single-level  
22 comparator 424. The multi-level comparator would determine a desired number of non-  
23 zero coefficients to discard per 8x8 block based on the value of the output of the

1 integrator 423. The maximum number of non-zero AC coefficients to keep for each 8x8  
2 block (i.e., the value of the parameter “k”), for example, would be determined by  
3 subtracting the number of non-zero AC DCT coefficients in the 8x8 block from the  
4 desired number to discard, and limiting this difference to no less than a predetermined  
5 fraction of the average number of non-zero AC coefficients per 8x8 block.

6

7           VII. Largest Magnitude Indices Selection For (Run, Level) Encoding Of A Block  
8           Coded Picture

9           As described above with reference to FIG. 15, one way of scaling original-quality  
10 MPEG-2 video to produce lower-quality MPEG video is to retain up to a certain number  
11 of largest-magnitude non-zero AC DCT coefficients and to truncate any remaining non-  
12 zero AC DCT coefficients from each 8x8 block of pixels. This was referred to as the  
13 FDSNR\_LM procedure. As shown and described with reference to FIG. 15, in step 262,  
14 the (run, level) event variable-length codes (VLCs) for each block are parsed and  
15 decoded to produce a set of quantization indices. In step 263, the quantization indices  
16 (for AC DCT coefficients) are transformed to quantized coefficient values, and in step  
17 264, the quantized coefficient values are sorted in descending order of their magnitudes.

18           In order to perform scaling in a computationally more efficient way, it is possible  
19 to eliminate the step 263 of transforming the quantization indices to quantized coefficient  
20 values by selecting largest magnitude quantization indices (for the AC DCT coefficients)  
21 instead of selecting the largest magnitude quantized coefficient values. This would make  
22 the scaling procedure more suitable for real-time applications, so long as there would not  
23 be a significant degradation in performance compared to the performance obtained by

1 selecting largest magnitude quantized coefficient values. For comparison purposes, the  
2 method of selecting the largest magnitude quantization indices will be referred to as  
3 “LMIS” (Largest Magnitude Indices Selection) and the method of selecting the largest  
4 magnitude quantized coefficient values will be referred to as “LMCS” (Largest  
5 Magnitude Coefficient Selection). It has been discovered that LMIS is not only  
6 computationally more efficient than LMCS but also LMIS provides an improvement in  
7 performance over LMCS in the rate-distortion sense.

8 FIG. 30 shows the LMIS procedure performed on an 8x8 pixel block for the case  
9 where no pivoting is used. (The use of pivoting will be discussed below with reference to  
10 FIG. 43.) In other words, a subroutine corresponding to the flowchart in FIG. 30 is called  
11 once for each series of variable-length codes corresponding to an 8x8 pixel block. In a  
12 first step 461, the differential DC coefficient representing variable-length code (VLC) is  
13 parsed and copied from an input bit stream of original-quality MPEG-2 video to an  
14 output bit stream of reduced-quality MPEG video. Then in step 462, all of the following  
15 (run, level) event variable-length codes (VLCs) are parsed and decoded until and  
16 including the first end-of-block (EOB) marker in the input bit stream. In step 463, the  
17 quantization indices are sorted in descending order of their magnitudes. Step 463 could  
18 use any of the sorting methods described above with reference to FIGs. 14 to 19. In step  
19 464, up to the first K indices of the sorted list are kept and the last 63-K indices of the  
20 sorted list are in effect set to zero. In other words, the last 63-K indices in the sorted list  
21 are not allowed to be represented by nonzero levels of (run, level) events in the output bit  
22 stream for the lower-quality MPEG video, but rather these last 63-K indices in the sorted  
23 list contribute to runs of zeros. In step 465, (run, level) event formation and entropy

1 encoding is applied to the new set of up to the first K indices in the sorted list. In the last  
2 step 466, the resulting VLCs are copied to the output bit stream until and including the  
3 end of block (EOB) marker.

4 The performance of LMIS would be equivalent to the performance of LMCS if  
5 the quantization of the coefficient values affected all coefficients to the same degree.  
6 However, MPEG-2 provides a visually weighted quantization matrix that modifies the  
7 quantization step size within a block. (See, for example, Section 7.4, Inverse  
8 Quantization, page 68, of the MPEG-2 International Standard ISO/IEC 13818-2  
9 Information technology - Generic coding of moving pictures and associated audio  
10 information: Video, 1996.) Therefore, DCT coefficients at higher frequencies with larger  
11 magnitudes may be mapped to indices with smaller values than DCT coefficients at lower  
12 frequencies with smaller magnitudes. Consequently, the ordering of the largest  
13 magnitude quantized coefficient values (produced by sorting the coefficient magnitudes  
14 in step 264 of FIG. 15) is not necessarily the same as the ordering of the largest  
15 magnitude quantization indices (produced by sorting index magnitudes in step 463 of  
16 FIG. 30).

17 If the video tends to have a predominant high spatial frequency content, then the  
18 coefficient matrix will tend to have larger values for the higher frequencies than the lower  
19 frequencies, and the visually weighted quantization matrix will cause the LMIS  
20 procedure to favor the selection of lower-frequency coefficients. In this case, the LMIS  
21 procedure will trade many of the indices with value one in the mid-to-high frequency  
22 range with value one (and sometimes 2 or 3) indices in the low frequency range. This  
23 swap of indices may result in lower picture signal-to-noise ratio (PSNR) than that

1 obtained via the LMCS procedure for the same number of retained AC coefficients.  
2 However, the bit savings achieved by the LMIS procedure are much more significant  
3 than the PSNR loss, and overall, the performance of LMIS is a lot better in the rate-  
4 distortion sense than that of LMCS. The indices that are retained by the LMIS procedure  
5 will not only shorten the run-lengths but also from the perceptual coding point of view, it  
6 will result in more pleasant images better matching subjectively/visually the Human  
7 Visual System's (HVS) low-pass resembling nature. That is not to say that mid-to-high  
8 frequency components are not important. It means simply that when the coefficient  
9 values are comparable to a certain extent, it is a better decision to keep the ones in the  
10 low-pass band both for bit savings and for achieving perceptual coding. Even though  
11 when two indices at different frequency channels have the same magnitude, the  
12 corresponding coefficient at the higher frequency channel being in general larger than  
13 that at the lower frequency channel, the difference does not justify the significantly  
14 higher bit-rate cost of coding the larger coefficient.

15 It may also be argued that LMIS is almost the same as the low-pass (LP) scaling  
16 described above with reference to FIG. 14. This is true for blocks of low-pass nature, and  
17 all three procedures (LMCS, LMIS, and LP) behave more or less the same when the  
18 signal power is concentrated in the low-pass band. However, when there is high signal  
19 power present in some other frequency band(s) as opposed to or in addition to the low-  
20 pass band, it is usually desirable to keep the signal components in those bands. While the  
21 LP scaling procedure can not achieve this, the LMCS procedure, being sensitive even to  
22 the smallest difference in the coefficient values, retains such bands even when the power  
23 in those bands is comparable to the power in the lower-frequency bands. Not only due to

1 high coding cost of indices in such bands but also due to the low-pass nature of the  
2 human visual system, we would like to favor the lower-frequency bands to higher  
3 frequency bands when the power levels are comparable. In many cases where the LMCS  
4 procedure favors the indices in higher frequency bands, it may be better to favor the  
5 indices in low-frequency bands due to extremely higher cost of coding indices in higher  
6 frequency bands. Two intermingled effects are present here. First, the indices at the  
7 higher frequencies are usually paired with longer run lengths, and secondly, the exclusion  
8 of indices in lower frequencies will result in even longer run lengths for the indices to be  
9 kept in those bands. Therefore, in the rate-distortion sense, the signal components in  
10 higher frequencies should be retained when they indeed represent some significantly  
11 strong image feature. LMIS does that very well. It balances the power and bit budget  
12 and hence results in better rate-distortion curves. LMCS, on the other hand, is strictly  
13 focused to obtaining the most signal power without any attention to the bit budget. If the  
14 bit budget were not a concern, LMCS could be a better choice. However, the problem  
15 domain dictates that these two factors must be balanced in an appropriate manner.

16 FIGS. 31 and 32 show performance comparisons between the LMCS and LMIS  
17 procedures for the case of no pivot insertion and the use of MPEG-2's default visually  
18 weighted quantization matrix having larger values for the higher frequencies than the  
19 lower frequencies for video. The performance improvement of LMIS over LMCS is less  
20 significant when pivot insertion is used, as described below, for avoiding escape  
21 sequences and reducing the bits needed for (run, level) coding.

22

23 VIII. Avoiding Escape Sequences In Coding of (Run, Level) Pairs

1           In view of the above, there have been described methods of efficient SNR scaling  
2       of video originally present in a high-quality and nonscalable MPEG-2 transport stream.  
3       To reduce the bandwidth of nonscalable MPEG-2 coded video, certain non-zero AC DCT  
4       coefficients for the 8x8 blocks are removed from the MPEG-2 coded video.

5           It is recognized that the largest magnitude coefficient selection (LMCS) procedure  
6       of FIG. 15, in theory, has a great potential to provide high-quality scaling. However, in  
7       practice, under the practical bit rate versus quality, peak signal-to-noise (PSNR) rate-  
8       distortion measure, the LMCS procedure has a performance problem. The source of the  
9       problem, in the most part, appears to be a mismatch (i.e. a non-compatibility), between  
10      the (run, level) event statistics generated by the LMCS procedure and the statistics that  
11      the MPEG-2 (run, level) VLC codebooks are designed for. This mismatch revealed itself  
12      by both the generation of a drastically increased number of escape sequences and also an  
13      increased tendency towards using less likely (according to the MPEG-2 base statistics)  
14      (run, level) symbols represented by longer code words.

15           There are two principal and coupled mechanisms leading to the problem. First of  
16       all, the LMCS procedure, by its very nature, retains the larger magnitude coefficients.  
17       Secondly, smaller magnitude coefficients in between the larger ones (to be retained) are  
18       discarded, leading to longer run lengths. The MPEG VLC codebooks are such that the  
19       larger the magnitude of indices and the longer the run lengths, the more bits are required  
20       to code them. Very often, the (run, level) pairs generated via LMCS fall out of the  
21       codebook, necessitating to resort to the costly (fixed 24 bits) Escape Sequence coding,  
22       which is a mechanism to code very rarely, in a statistical sense, occurring (run, level)  
23       pairs.

1 As described above with reference to FIGS. 20 and 21, one way of avoiding  
2 escape sequences from the LMCS procedure was to include a non-zero, non-qualifying  
3 AC DCT coefficient in the (run, level) coding. As described below, this method can be  
4 extended to avoid escape sequences or reduce the number of bits used in the variable-  
5 length coding by introducing indices of magnitude 1 into coefficient channels  
6 corresponding to zero-valued AC DCT coefficients in the original-quality MPEG-2 coded  
7 video. In any case, a quantization index introduced into the coding for the reduced-  
8 quality MPEG coded video for the purpose of avoiding an escape sequence or reducing  
9 the number of bits for variable-length coding will be referred to as a "pivot index." The  
10 pivot indices, when used jointly with the LMCS or LMIS procedures, effectively change  
11 the original statistics of the (run, level) symbols generated by the baseline LMCS or  
12 LMIS procedures. In fact, the pivot technique, as described below, is useful in  
13 combination with any encoding or scaling technique producing (run, level) codes that  
14 deviate from the normal MPEG statistics by having a greater than normal frequency of  
15 escape sequences and an increased probability mass on (run, level) symbols with  
16 relatively long codewords.

17 The objective of inserting pivot indices is to break the long run-lengths of zero  
18 valued AC DCT coefficients when such a split leads to a savings in the number of bits  
19 required for encoding. The basic underlying principle is that if preserving a non-  
20 qualifying smaller magnitude coefficient normally to be discarded by the LMCS (or the  
21 LMIS) procedure requires fewer bits to encode both itself and the following larger  
22 magnitude coefficient which was originally to be retained, then one can shoot two birds  
23 with one stone by preserving this non-qualifying coefficient. That is to say, not only a bit

1 savings is achieved but also the quality, i.e. the PSNR measure, improves due to the  
2 inclusion of one more genuine coefficient.

3 The following example illustrates the basic underlying principle. Assume that in  
4 a sample quantized 8x8 coefficient block, a partial listing of the indices ordered  
5 according to the employed zigzag scan order is given as (... , 9, 0, 3, 6, ...). Assume  
6 further that the LMCS algorithm decides to retain the coefficients associated with the  
7 indices 9 and 6 but not 3. Then, the index 3 will be treated as zero resulting in a (run,  
8 level) pair of value (2, 6). The symbol (2, 6) is not allocated a particular variable length  
9 codeword in the codebook and hence its encoding requires the use of an Escape Sequence  
10 of 24 bits. However, if we decide to retain the index 3 as well, then two alternate (run,  
11 level) pairs, namely (1, 3) and (0, 6), need to be encoded instead. Since the encoding of  
12 the latter two symbols requires 15 bits in total, 9 bits are saved with respect to the first  
13 alternative. Also, the inclusion of the index 3 contributes in a positive sense to reduce the  
14 power of the reconstruction error. Here, the index 3 is the pivot index. This technique is  
15 the first version, called Pivot 1, of a class of pivot techniques summarized in FIG. 33. As  
16 shown in the first step 481 of FIG. 33, the Pivot-1 technique selectively retains genuine  
17 non-qualifying non-zero AC coefficients in order to avoid escape sequences.

18 The primary motivation for preserving the index 3 in the above example is to save  
19 bits by avoiding the generation of an escape sequence which is the most inefficient way  
20 of encoding quantized coefficient data in MPEG-2. The marginal improvement in the  
21 PSNR came as a side benefit. An analysis of both of the (run, level) VLC codebooks  
22 (Table 0 and Table 1) employed by MPEG-2 reveals the fact that for a fixed value of the  
23 run-length, the lengths of the codewords are always defined by a monotonic non-

1 decreasing function of the level. Since the marginal SNR improvement provided by the  
2 pivot index is of secondary significance, why not, then, change the value of the pivot  
3 index to plus one or minus one depending on the sign of its original value and achieve a  
4 further savings in bits? Going back to our previous example, when we apply this idea to  
5 the original two (run, level) pairs, we get the symbols (1, 1) and (0, 6) the encoding of  
6 both of which requires 11 bits instead of 15. It should be also noted that, even though to  
7 a lesser extent with respect to its preservation in its original value, the inclusion of an  
8 index with magnitude one and the correct sign, still contributes positively to the quality  
9 (i.e., PSNR) as compared to the case of its total elimination. This version will be called  
10 the Pivot-2 technique. As shown in step 482 of FIG. 33, the Pivot-2 technique reduces  
11 the level magnitudes of the retained non-qualifying non-zero AC coefficients to a value  
12 of one in order to eliminate more escape sequences and to reduce the number of bits for  
13 (run, level) encodings.

14 A third and final version of the pivot techniques, called Pivot-3, involves inserting  
15 a pivot of level magnitude 1 for a level zero coefficient in the transformation of the  
16 original high-quality 8x8 pixel block into the lower quality version. (See step 483 in  
17 FIG. 33.) In effect, the pivot is noise that is inserted into the picture to obtain a more than  
18 compensating benefit of reducing the number of bits to encode the picture. Moreover, the  
19 objective reduction in PSNR due to inserting the noise-like pivots is masked subjectively  
20 by inter-coefficient contrast masking and in many cases is not visible to a casual human  
21 observer.

22 Consider first the relation of the escape sequence count with the number of  
23 preserved coefficients in each block. FIGs. 34 and 35 show the plots of the average

number of escape sequences per frame as a function of the number of LMCS coefficients retained in each block for various quantization levels (qsv) at the input of the LMCS processing, and with or without the various pivoting mechanisms. The data for these plots was produced by averaging over representative frames from the three standard test sequences (namely, Susie, Flower Garden and Football). In these plots and for a fixed input quantization level, the general trend which is actually to a significant extent common to all four (including Pivot-3) illustrated different LMCS and pivoting combinations, is as follows. When the baseline LMCS algorithm is configured to preserve only a few largest magnitude coefficients, the number of escape sequences generated is quite high. This is not only because the preserved indices are (most of the time) the largest magnitude indices of all but also because of the fact that the few number of preserved indices achieve only a very sparse sampling of the 8 x 8 coefficient grid leading to significantly increased run-lengths. For indices with magnitudes greater than 5, even a run-length as small as 3 will result in an escape sequence. As more indices (associated with largest magnitude coefficients) are preserved, the number of escape sequences continues to increase albeit a decreasing slope (i.e., a decreasing rate of generation) with each unit increase in the number of preserved AC coefficients per block. After a relatively small number of preserved indices (in the range from 5 to 10), the number of escape sequences starts declining with further increase in the number of preserved indices.

There are two mechanisms contributing to the observed relation of the escape sequence count with the number of preserved coefficients in each block. The first is the decrease in the magnitudes of the smallest indices which made it to the list of preserved

1 indices due to increased nonzero coefficient allowance per block. This magnitude  
2 decrease, decreases the likelihood of their generating escape sequences. But even more  
3 important and influential than this observation is the fact that the inclusion of a steadily  
4 increasing number of nonzero indices leads to a denser population of the 8x8 coefficient  
5 block, effectively breaking long runs of zeros between two large magnitude coefficients.  
6 This will lead to a shortening of the run-length components of the symbols to be encoded  
7 and hence a reduction in the number of escape sequences generated as well as an  
8 increased tendency towards using (run, level) symbols associated with shorter VLCs.

9 Even after the improvement achieved by the Pivot-2 procedure, there still remains  
10 quite a significant number of escape sequences. The effectiveness of and therefore the bit  
11 savings associated with the Pivot-2 procedure are limited since, more often than not,  
12 either there are no genuine candidate pivot indices between two qualifying largest  
13 magnitude coefficient indices or it is not feasible to use a genuine pivot index since it  
14 requires more encoding bits to include it owing to the locations and/or magnitudes of  
15 both the pivot and the qualifying largest magnitude coefficient. The nature of the  
16 coefficient selection implemented by the LMCS procedure and a very interesting human  
17 visual system masking behavior pertinent to DCT basis images open the way to another  
18 enhancement in the pivot index insertion framework.

19 The sensitivity of the human visual system to different DCT basis images is  
20 different. Here, the sensitivity is defined as the reciprocal of the smallest magnitude (i.e.,  
21 the threshold amplitude) of the basis image which enables its detection by human  
22 observers. This sensitivity is also dependent on various other factors. Among these are  
23 the viewing conditions such as the ambient luminance, and the display parameters such as

1 the display luminance (the mean background intensity) and the visual resolution of the  
2 display (specified in terms of the display resolution and the viewing distance). However,  
3 more important for bit reduction purposes are the so called image-dependent factors  
4 which model the mechanisms generated by the simultaneous presence of more than one  
5 basis image.

6 One very significant image-dependent factor influencing the detectability of DCT  
7 basis images is the effect of contrast masking, as described in Andrew B. Watson, Joshua  
8 A. Solomon, A. J. Ahumada Jr. and Alan Gale, "DCT Basis Function Visibility: Effects  
9 of Viewing Distance and Contrast Masking," in B. E. Rogowitz (Ed.), Human Vision,  
10 Visual Processing, and Digital Display IV (pp. 99-108). Bellingham, WA: SPIE, 1994.  
11 This paper includes the following basic model of contrast masking:

$$m_T = t_T \max \left[ 1, \left| \frac{c_T}{t_T} \right|^{w_T} \right],$$

12 where:

13 the subscript  $T$  implies association with the spatial frequency  $T = (u, v)$ ,  $u, v = 0, 1, \dots, 7$ ,  
14 defined on the basis of the indices of the corresponding DCT coefficient;  
15  $c_T$  is the given DCT coefficient;  
16  $t_T$  is the corresponding absolute threshold;  
17  $w_T$  is an exponent that lies between 0 and 1; and  
18  $m_T$  is the masked threshold.

19  
20 In the above equation,  $m_T$  defines the maximum extent of deviation from the  
21 coefficient's original value  $c_T$  which will not be detected by a typical human observer,

1 when the correspondingly weighted basis image is displayed. It is easy to see from this  
 2 model that, typically, sensitivity to quantization error in a particular DCT coefficient,  
 3 decreases with the magnitude of that coefficient due to the increased masked threshold.  
 4 Note that this first order model of contrast masking describes the sensitivity to a  
 5 particular coefficient's quantization error as being independent of the magnitudes of all  
 6 the other coefficients except for the DC coefficient. However, there is evidence to the  
 7 contrary, which indicates that sensitivity to a particular coefficient's quantization error is  
 8 affected by the magnitudes of other coefficients (i.e., inter-coefficient contrast masking)  
 9 as described through the following model:

$$m_T = t_T \max \left[ 1, \left| f[T, M] \frac{c_M}{t_T} \right|^w \right],$$

$$f[T, M] = \exp \left[ -\pi \frac{\|T - M\|^2}{\zeta_T^2} \right],$$

$$\zeta_T = \zeta \max[1, \|T\|].$$

10

11 In the above model:

12 the subscript  $T$  implies association with the spatial frequency  $T = (u, v)$ ,  $u, v =$   
 13  $0, 1, \dots, 7$ , defined based on the indices of the corresponding DCT coefficient;

14 the subscript  $M$  implies association with the spatial frequency  $M = (x, y)$ ,  $x, y =$   
 15  $0, 1, \dots, 7$ , defined based on the indices of the corresponding DCT coefficient;

16  $c_T$  is the DCT coefficient associated with the given test DCT basis image;

17  $t_T$  is  $c_T$ 's corresponding absolute threshold;

18  $c_M$  is the DCT coefficient associated with the given masking DCT basis image;

1            $w$  is an exponent that lies between 0 and 1;  
2            $f[T, M]$  is a positive, frequency-dependent scaling factor; and  
3            $m_T$  is the masked threshold for  $c_T$  due to the presence of  $c_M$ .

4

5           Observe that  $f[T, M]$  assumes its maximum value of 1 when  $T = M$ . In this case,  
6           this latter improved model reduces to the first order model described above.  $f[T, M]$   
7           reflects the sensitivity of  $c_T$ 's detection to the presence of a masking coefficient  $c_M$  at the  
8           frequency  $M$ . The larger  $f[T, M]$  is, the stronger is the masking influence. (The precise  
9           extent of masking generated by  $c_M$  is also dependent on the ratio  $c_M / t_T$ ). The second  
10          equation in the description of the improved model defines  $f[T, M]$  as a radially  
11          symmetric Gaussian function parameterized through a single parameter  $\zeta$  defined in the  
12          third equation of the same description. It is interesting to note that the bandwidth of  $f[T,$   
13           $M]$  increases in proportion to (the  $L^2$  norm of the spatial) frequency except for the DC  
14          coefficient. This, in particular implies a reduced sensitivity to (equivalently an easier  
15          masking of) high-frequency coefficients. The three fundamental parameters of the  
16          refined model, namely  $w$ ,  $t_T$  and  $\zeta$  are determined through a least squares estimation  
17          method on empirical data.

18          The relation of the contrast masking phenomenon to the first two generations of  
19          the pivot index methodology is easy to conclude. The qualified largest magnitude  
20          coefficients selected by the LMCS algorithm typically have the potential to generate a  
21          strong masking effect in their close vicinity in the frequency domain due to an increased  
22           $c_M / t_T$  ratio. Furthermore, the nature of both zigzag scans (i.e., achieving a slowly  
23          changing frequency content) lead to the insertion of pivot indices which are (most of the

time) very close to their respective qualified LMCS coefficients in the frequency domain.  
This in return leads to a smaller value for the metric  $\|T - M\|$  in the exponent of  $f[T, M]$ , increasing its value too. (In case of alternate zigzag scan, there are a few cases of potentially having a somewhat larger frequency difference between the frequency of the preserved index and the frequency of the pivot index compared to the case of the default zigzag scan. Yet, these increased differences when they are realized, only weaken the extent of inter-coefficient contrast masking but do not eliminate masking altogether.) Given the two effects of large magnitude qualified coefficients generating a strong masking in their frequency domain neighborhoods and the pivot indices almost always being placed very close to the qualified coefficients, by moving from the first version (Pivot-1) to the second version (Pivot-2) of the pivot index methods, we achieve a further savings in the number of encoding bits at the expense of a marginal PSNR reduction which is visually masked.

We will now carry this idea one step further and consider introducing an artificial pivot when genuine ones are missing or inefficient for their purpose. Such an action will be taken only when it is beneficial to do so. For the best savings in the required number of encoding bits and for the least additional distortion, the pivot index should be a plus one or a minus one. There are several interesting observations and conclusions regarding this class of the pivot index technique and one of its possible implementations.

First, a pivot index should be placed in the position immediately preceding the position of the largest magnitude index in the adapted zigzag scanning order. This position is either the only position to get any savings or the position for the largest savings among all alternative positions. Consequently, a (run, level) symbol (n, M) is

1 transformed into the cascade combination of symbols (n-1, 1) and (0, M). There are a  
2 few minor exceptions to this rule, and these exceptions can be encoded in a special table  
3 or tested for prior to a table lookup, as described below. A corollary to this observation is  
4 that the implementation complexity is very low since there is no extensive decision and  
5 search processes involved as to where to place the pivot.

6 Second, the decision as to whether placing a pivot will help or not is as simple as  
7 a small table lookup. Given that we have to code the (run, level) symbol (n, M), we  
8 immediately know that the symbols (n-1, 1) and (0, M) must be coded instead if a pivot is  
9 to be employed. The analysis of savings associated with this type of symbol  
10 transformation can be performed once off-line for all possible (n, M) pairs and a decision  
11 table to be indexed by n and M, can be generated to make the decision in the form of a  
12 Yes (1) or No (0) answer. For both of MPEG-2's VLC tables (Table 0 and Table 1) if the  
13 run-length (n) is equal to 0 or greater than 32 or the level (M) is greater than 40 in  
14 magnitude, the above proposed pivot technique cannot help. Therefore, the required  
15 table size is 32x40 bits =160 Bytes.

16 Third, the degrading influence of such pivots on the reconstruction quality with  
17 respect to the PSNR metric is marginal since they are used in association with the LMCS  
18 approach. More importantly, as discussed above, the distortion introduced by the pivots  
19 is perceptually masked to a large extent due to inter-coefficient contrast masking.  
20 Moreover, as will be described below with reference to FIGs. 41 to 44, if a decoder is  
21 aware that the Pivot-3 technique is employed by an encoder (or a transcoder), then the  
22 decoder can, with a high accuracy, distinguish genuine indices with value (+/-) 1 from the

1 inserted, noise-like pivot indices, and therefore the decoder can remove most of the  
2 inserted, noise-like pivot indices to avoid any significant degradation in PSNR.

3 With reference to FIG. 36, there is shown a sequence of DCT coefficients ... $C_i$   
4 ... $C_j C_k$ ... in the coefficient scan order giving rise to a sequence of (run, level) symbols  
5 for encoding an 8x8 block of pixels. In particular, the run length for the (run, level)  
6 symbol to be used for encoding the coefficient  $C_k$  is determined by the number R of  
7 consecutive AC coefficients having a zero level and immediately preceding the  
8 coefficient  $C_k$  in the scan order. In this example, the coefficients  $C_i$  and  $C_k$  are  
9 superscripted with asterisks to indicate that they have non-zero levels. The Pivot-3  
10 technique decides whether or not the (run, level) coding for each non-zero AC  
11 coefficient, such as the coefficient  $C_k$ , should be modified by changing the level of an  
12 immediately preceding coefficient  $C_j$ , in the scan order, from level 0 to a level of  
13 magnitude 1 (i.e., from 0 to a level of +1 or -1).

14 In order to decide whether or not the (run, level) coding for each non-zero AC  
15 coefficient should be modified by changing the level of an immediately preceding  
16 coefficient in the scan order, a lookup operation can be performed on a two-dimensional  
17 pivot table having a respective entry for each possible run length and level magnitude.  
18 As shown in FIG. 37, for example, the pivot table may include 64 rows for each possible  
19 encoded run length from 0 to 63, and 2048 columns for each possible encoded level  
20 magnitude from 1 to 2048. Inspection of such a pivot table, however, shows that no pivot  
21 should (or can) be inserted for a run length of zero, a run length greater than 32, or a level  
22 magnitude greater than 40. Therefore, considerable table memory may be saved by only  
23 storing a partial pivot table (497) having 32 rows and 40 columns.

1           With reference to FIG. 38, there is shown a first sheet of a flowchart for the Pivot-  
2       3 procedure. In a first step 501, scanning of a stream of indices begins in a next block. If  
3       the end of the stream is reached, as tested in step 502, then the procedure is finished.  
4       Otherwise, execution continues to step 503, to get the next index to be coded. If the end  
5       of the block is reached, as tested in step 504, then execution loops back to step 501.  
6       Otherwise, execution continues to step 505 to determine the next (run, level) pair (M, N).  
7       In step 506, the run and level values are used to lookup a pivot table. If the pivot table  
8       indicates that a pivot should be inserted, then execution continues from step 507 to step  
9       508 in FIG. 39. Otherwise, execution branches from step 507 to step 513 in FIG. 39.

10           In step 508 of FIG. 39, a lookup is performed on the original index block for the  
11       immediately preceding location in the scan order to see if there was a genuine index to be  
12       discarded; in other words, an index for a non-zero, non-qualifying AC coefficient in the  
13       original-quality picture. If there is such an index, as tested in step 509, then execution  
14       continues to step 510 to lookup the VLC table for a run length of M-1 and a level equal to  
15       one in magnitude with a sign the same as the sign of the found index, in order to insert a  
16       pivot index having the corresponding VLC code. Then, in step 512, the VLC table is  
17       looked up for a run length of zero and a level of N, to re-code the variable-length code for  
18       the index obtained in step 503. In other words, instead of coding the variable-length code  
19       for the (run, level) of (M, N), a savings in bits is achieved by coding the variable-length  
20       code for (M-1, SIGN(INDEX)\*1) followed by the variable-length code for (0, N). From  
21       step 512, execution loops back to step 503 in FIG. 38. If there is no genuine non-zero,  
22       non-qualified index as tested in step 509, then execution continues to step 511 to lookup  
23       the VLC table for a run length of (M-1) and a level equal to one in magnitude with

1 always a positive sign, in order to insert a pivot index having the corresponding VLC  
2 code. After step 511, the execution continues to step 512. In other words, instead of  
3 coding the variable-length code for the (run, level) of (M, N), a savings in bits is achieved  
4 by coding the variable-length code for (M-1, 1) followed by the variable-length code for  
5 (0, N). It should be noted that for the artificial pivot index inserted in step 511 we  
6 arbitrarily but uniformly chose a positive sign, which to some extent supports the  
7 discrimination of artificial pivot indices from similar-looking genuine indices at the  
8 decoder. If a pivot is not to be inserted, then in step 513 of FIG. 39, the VLC table is  
9 looked-up for a run length of M and a level of N, in order to code the variable-length  
10 code for (M, N). Execution loops from step 513 back to step 503 in FIG. 38.

11 With reference to FIG. 40, there is shown a flowchart of a subroutine that  
12 simulates a lookup in the pivot table of FIG. 37 by performing some comparisons and, if  
13 necessary, performing a lookup of the partial pivot table (497 in FIG. 37). In a first step  
14 521 in FIG. 40, if the run length is zero, then execution branches to step 522, to return an  
15 indication that a pivot is not to be inserted. If the run length is not zero, then execution  
16 continues from step 521 to step 523. In step 523, if the run length is greater than 32,  
17 execution branches to step 522, to return an indication that a pivot is not to be inserted. If  
18 the run length is not greater than 32, then execution continues from 523 to step 525. In  
19 step 525, the level magnitude is computed as the absolute value of the level. In step 526,  
20 if the magnitude is greater than 40, then execution branches to step 522 to return an  
21 indication than a pivot is not to be inserted. In step 526, if the magnitude is not greater  
22 than 40, execution continues to step 527. In step 527, a lookup is performed upon the  
23 partial pivot table.

1       Following is a listing of the partial pivot table, for VLC coding according to the  
2       MPEG-2 VLC coding Table One:

3

4       % Partial pivot table description.

5       % value 0 => Pivot is not to be used. Code as it is.

6       % value -1 => Use a pivot only if there is already a non-zero coefficient in the  
7       pivot position.

8       % value of 1 => Use a pivot.

9       % "zeros(n)" denotes a row vector of "n" zeros; "ones(n)" denotes a row vector of  
10      "n" ones.

11

12      zeros(2) -1 -1 -1 ones(10) zeros(3) ones(22):

13      zeros(1) -1 -1 ones(37);

14      zeros(2) ones(38);

15      zeros(2) ones(38);

16      zeros(2) ones(38);

17      zeros(2) ones(38);

18      0 1 ones(29) zeros(9);

19      0 1 ones(29) zeros(9);

20      0 1 ones(29) zeros(9);

21      0 1 ones(29) zeros(9);

22      0 1 ones(29) zeros(9);

23      0 1 ones(13) zeros(25);

1           0 1 ones(13) zeros(25);  
2           0 1 ones(13) zeros(25);  
3           0 1 ones(13) zeros(25);  
4           0 1 ones(13) zeros(25);  
5           0 1 ones(13) zeros(25);  
6           0 1 ones(13) zeros(25);  
7           0 1 ones(13) zeros(25);  
8           0 1 ones(13) zeros(25);  
9           0 1 ones(13) zeros(25);  
10          0 1 ones(13) zeros(25);  
11          0 1 ones(13) zeros(25);  
12          0 1 ones(13) zeros(25);  
13          0 1 ones(13) zeros(25);  
14          0 1 ones(13) zeros(25);  
15          0 1 ones(13) zeros(25);  
16          0 1 ones(3) zeros(35);  
17          0 1 ones(3) zeros(35);  
18          0 1 ones(3) zeros(35);  
19          0 1 ones(3) zeros(35);  
20          0 1 ones(3) zeros(35);  
21  
22          The partial pivot table listed above has a few entries of value -1. For each of  
23          these entries, the insertion of a pivot results in exactly the same number of bits that would

1       be needed if a pivot were not inserted. Therefore, in these cases a pivot should be  
2       inserted only if there is a genuine nonzero index (i.e., an index having a non-zero level in  
3       the original picture) at the pivot location. Pivot inclusion in the location of such an  
4       already existing nonzero index will help to improve the signal quality without using any  
5       more bits for (run, level) encoding. However, when such a pivot insertion is made, in  
6       accordance with the general pivot insertion rules, even if the level magnitude in the  
7       original picture is greater than one, the level magnitude of the inserted pivot should be set  
8       to one, and the sign of the level of the inserted pivot should be the same as the sign of the  
9       level in the original picture.

10           Storage of the -1 values in the memory allocated to the table would unduly  
11       increase the amount of memory needed for the table. Instead, these few entries can be  
12       coded in the table look-up process as follows:

13

14           % lookup of the partial pivot table

15           START     IF (RUN = 1) THEN GOTO 100

16                   IF (RUN = 2) THEN GOTO 200

17           50        PPTV <= PPT(RUN, MAG)

18                   RETURN

19           % table lookup returns a value of PPTV = 0 or 1

20           100       IF (MAG<3) THEN GOTO 50

21                   IF (MAG>5) THEN GOTO 50

22           150       PPTV <= -1

23                   RETURN

1                   % returns a value of PPTV = -1  
2       200           IF (MAG<2) THEN GOTO 50  
3                   IF (MAG>3) THEN GOTO 50  
4                   GOTO 150  
5

6                   Returning now to FIG. 40, in step 528, if the partial pivot table value (PPTV), for  
7     the row = RUN and column = MAG, has a value of zero, then execution branches to step  
8     522 to return an indication that no pivot should be inserted. Otherwise, execution  
9     continues from step 528 to step 529. In step 529, if the partial pivot table value (PPTV)  
10   is equal to 1, then execution branches to step 524 to return an indication that a pivot  
11   should be inserted. Otherwise, for the case of PPTV = -1, execution continues to step  
12   530. In step 530, execution branches depending on whether or not there is a non-  
13   qualifying coefficient (i.e., an AC coefficient having a non-zero level in the original  
14   picture) in the pivot position. If so, then execution branches to step 524 to return an  
15   indication that a pivot should be inserted. If not, then execution branches to step 522 to  
16   return an indication that a pivot should not be inserted.

17                  FIG. 41 shows the Pivot-3 method of inserting pivot indices during the encoding  
18     or transcoding process and partial removal of the pivot indices during the decoding  
19     process. Transform coefficients are obtained from an original block-coded picture 541.  
20     The encoding or transcoding process includes the Pivot-3 method 542 of inserting noise,  
21     in the form of pivot indices, to reduce the number of bits for (run, level) coding of the  
22     transform coefficients. The reduction in the number of bits facilitates the transmission or  
23     storage 543 of the (run, level) coded transform coefficients. The decoding process 544

1 includes the partial removal of the noise (i.e., the pivot indices) by removal of possible  
2 pivot indices not likely to occur in the original block-coded picture. The resulting  
3 transform coefficients are then used in a decoding process 545 of producing a  
4 reconstructed block-coded picture.

5 In order to facilitate the removal of possible pivot indices not likely to occur in an  
6 original block-coded picture, the Pivot-3 encoding method can be adjusted depending on  
7 whether or not the decoder will attempt removal of pivots. For example, as shown in  
8 FIG. 42, if the decoder will not attempt removal of pivots, as tested in step 551, then the  
9 process of encoding or transcoding will insert artificial pivots having a level of +1 or -1  
10 selected in a substantially random fashion, as shown in step 552. (For example, step 511  
11 of FIG. 39 would be modified to lookup the VLC table for either (M-1, 1) or (M-1, -1)  
12 selected by a pseudo-random number generator function.) However, if the decoder will  
13 attempt removal of pivots, then the level (more precisely the sign) of the artificial pivot  
14 indices should be selected in a convenient way such that the decoder will know whether  
15 or not a +1 or -1 is selected for the level of any pivot index inserted into the (run, level)  
16 coding. The most convenient way to perform such a selection of the pivot index level is  
17 to insert pivot indices all having the same level of either +1 or -1, such as a level of +1 as  
18 shown in step 553. (This is also what is shown in step 511 of FIG. 39.) Therefore,  
19 during the decoding process, all indices with value -1 (on average forming 50% of the  
20 genuine indices with magnitude 1) are certainly known to be genuine indices that should  
21 not be removed.

22 For distinguishing the genuine indices from the pivots within the set of indices  
23 with value +1, one can apply several rules. A (+1) level index immediately followed by a

1       (+/-) 1 level index or an end-of-block (EOB) symbol in the scan order is a genuine index.  
2       A (+1) level index which is not immediately followed by another nonzero index in the  
3       scan order is a genuine index. If a (+1) level index and the nonzero index immediately  
4       following it together form an (n, +1) and (0, M) symbol pair, (i.e., a potential pivot  
5       location encountered), still many cases exist in which depending on the values of the  
6       tuple (n, M), we can identify with certainty whether that (+1) level index is a genuine  
7       index or a pivot. For example, if M = 32 and n > 6, we know with certainty that the (+1)  
8       level index is not a pivot index. That is, (7, 32), (8, 32), ... (31, 32) are the set of tuples  
9       for which the (+1) level index is a genuine index. It is because for these tuples the total  
10      number of bits to code both (n, +1) and (0, 32) is not less than 24 bits required to code  
11      (n+1, M).

12           FIG. 43 shows in greater detail the procedure of attempted pivot removal during  
13          decoding. In a first step 561, the decoder gets the next (run, level) pair. In step 562, if  
14          the end of the current encoded block is reached, then the procedure is finished.  
15          Otherwise, execution continues to step 563. In step 563, if the coefficient encoded by the  
16          (run, level) pair is not possibly a pivot, then execution branches to step 565 to accept the  
17          coefficient. Otherwise, if the coefficient is possibly a pivot inserted by the Pivot-3  
18          technique, then execution continues to step 564. In step 564, if the coefficient is not  
19          likely to be a pivot inserted by the Pivot-3 technique, then execution branches to step 565  
20          to accept the coefficient. Otherwise, if the coefficient is likely to be a pivot inserted by  
21          the Pivot-3 technique, then execution continues to step 566 to reject the coefficient. After  
22          step 565 or 566, execution loops back to step 561 to process the next (run, level) pair.

1 FIG. 44 shows further details of the process of determining whether or not an  
2 index is possibly a pivot (corresponding to step 563 in FIG. 43) and whether or not an  
3 index is likely to be a pivot (corresponding to step 564 in step FIG. 43). In a first step  
4 571 of FIG. 44, if the level is not equal to 1, execution branches to accept the coefficient.  
5 Otherwise, execution continues to step 572. In step 572, the decoding process looks  
6 ahead to the immediately following symbol in the (run, level) symbol stream. If this  
7 immediately following symbol is an end-of-block (EOB) symbol as tested in step 573,  
8 then execution branches to accept the coefficient. Otherwise, execution continues to step  
9 574. In step 574, if the run length of the immediately following symbol is not zero, then  
10 execution branches to accept the coefficient. Otherwise, execution continues to step 575.  
11 In step 575, the magnitude of the level of the immediately following symbol is computed.  
12 Then in step 576, if the magnitude of the immediately following symbol is not greater  
13 than one, execution branches to accept the coefficient. Otherwise, execution continues  
14 from step 576 to step 577. In step 577, the decoder computes the run length that the  
15 immediately following symbol would have had if the coefficient (i.e., the possible pivot)  
16 were rejected. This is done by incrementing the run length by one. Then in step 578, the  
17 decoder looks up the pivot table with the run length (from step 577) and the magnitude  
18 (from step 575) in order to determine what the encoder would have done if the coefficient  
19 had a zero level in the original picture. In step 579, if the encoder would not have  
20 inserted the coefficient as a pivot, then execution branches to accept the coefficient.  
21 Otherwise, if the encoder would have inserted the coefficient as a pivot, then the  
22 coefficient is rejected as it is likely to have been inserted by the encoder.

1        It should be noted that steps 577, 578 and 579 could be omitted, in order to reject  
2        the coefficient if step 576 finds that the magnitude of the immediately following symbol  
3        has a level magnitude greater than one. Since the possible adverse effects of the pivots  
4        on the perceived picture quality is so small, it may not be worthwhile to perform steps  
5        577 to 579. However, if the table memory is allocated anyway for encoding purposes, for  
6        example in a transceiver application or a picture storage and retrieval application, then  
7        the cost of performing steps 577, 578, and 579 would be minimal.

8        It should also be noted that the pivot table used in step 578 could be slightly  
9        different from the pivot table used for encoding, in order that the pivot table for decoding  
10      and rejecting possible pivots could take into account the probability that, when step 578  
11      is reached, a (RUN, MAG) pair would occur in the original picture, and the pivot table in  
12      step 578 would indicate the insertion of a pivot point, yet a pivot would not have been  
13      inserted by the encoder or transcoder because the original picture would include an  
14      immediately preceding coefficient of level = 1. Such slight differences in the tables  
15      could occur for (RUN, MAG) pairs having both a short run length and a small magnitude,  
16      and they could be found by encoding a series of test pictures and computing a histogram  
17      indicating, for each possible (RUN, MAG) pair, the percentage of the time that the  
18      procedure of FIG. 44 rejects a coefficient found in the original picture (run, level) coding  
19      when steps 577 to 579 are reached. If this percentage is greater than 50%, then the pivot  
20      table entry for the (RUN, MAG) pair should be changed so that this percentage would  
21      become less than 50%.

22        Although the pivot insertion procedures provide the most significant  
23        improvements when used in conjunction with the LMCS scaling procedure, the pivot

1 insertion procedures may also provide substantial reductions in the bits required for  
2 encoding when used together with the LMIS scaling procedure or other scaling or  
3 encoding procedures. FIG. 45, for example, shows a flow diagram for using both the  
4 LCMS procedure and the LMIS procedure in combination with the pivot insertion  
5 techniques for both transcoding and encoding. For transcoding, a (run, level) encoded bit  
6 stream for a picture is produced by a high resolution encoder 581 or a low resolution  
7 encoder 582. A decoder 583 decodes the (run, level) encoded bit stream.

8 For using the LMCS procedure, a de-quantizer 584 de-quantizes the levels to  
9 produce corresponding coefficient values. Coefficient selection 585 is performed  
10 according to the LMCS procedure, and pivot insertion 586 to reduce the number of bits  
11 for encoding is performed on the selected coefficients, to produce a (run, level) encoded  
12 picture.

13 Coefficient selection 587 by the LMIS procedure is performed on the decoded  
14 (run, level) information, and the pivot insertion 586 is performed on the selected  
15 coefficients to produce a (run, level) encoded picture.

16 When using the LMCS or LMIS procedures with pivot insertion during encoding,  
17 a DCT encoder 588 produces DCT coefficient values for 8x8 pixel blocks in the picture.  
18 Coefficient selection 589 by LMCS is followed by coefficient quantization to produce  
19 corresponding level values, which are used during the pivot insertion 586 to produce the  
20 (run, level) encoded picture. For the LMIS procedure, the DCT coefficient values from  
21 the DCT encoder 588 are processed by a quantizer 591 to produce a series of  
22 corresponding level values, and coefficient selection 592 by LMIS produces a subset of  
23 these level values for the pivot insertion 586 to produce the (run, level) encoded picture.

1      The 8x8 bank of quantizers 591 may have quantization step sizes that are not uniform  
2      within the 8x8 block of DCT coefficients, for example, so that the higher frequency  
3      coefficients in each block are quantized with a larger step size than the lower frequency  
4      coefficients in the block.

5           Although the LMCS, LMIS, and pivot insertion methods have been described  
6      with respect to reducing the number of bits for encoding pictures which are MPEG-2  
7      video frames, it should be understood that the methods have general applicability to  
8      reducing the number of bits for (run, level) encoding regardless of the information  
9      represented by coefficients that are encoded. For example, the methods are directly  
10     applicable to scaling and encoding of individual pictures encoded by JPEG, which also  
11     run-length encodes DCT coefficients for 8x8 pixel blocks. The methods are also  
12     applicable to the use of other transform encoding techniques such as Fourier transform or  
13     wavelet transform techniques, and the encoding and compression of one-dimensional  
14     signals, such as audio signals.